Red Hat | aws

Integrate

Red Hat OpenShift Service on AWS with AWS services

A step-by-step guide

Contents

Chapter 1

Talk to AWS and Red Hat

If you are considering **Red Hat® OpenShift® Service on AWS (ROSA)**, we want to connect and talk with you. While this e-book provides guidance on getting started and integrating cloud-native services with OpenShift Service on AWS, Red Hat and Amazon Web Services (AWS) offer many more resources to help you take your experience further. We want to make sure that OpenShift Service on AWS is the platform that meets your application and innovation needs.

OpenShift Service on AWS was created for one simple reason: customers like you asked for it. More than ever before, our joint customers—enterprises and organizations of all sizes—are deploying Red Hat's portfolio on AWS.

Red Hat OpenShift has been fully supported as a self-managed offering on AWS for years, providing a customizable option for organizations that want more control over their environment. However, setup, deployment, and Day 2 management require both expertise and time that some customers may not have.

A growing number of organizations have successfully adopted OpenShift Service on AWS—the fully managed offering of Red Hat OpenShift on AWS. These customers spend far less time on setup and Day 2 operations and more time focusing on their applications.

We created this guide based on our own hands-on experience to provide you with best practices for integrating AWS-native services with OpenShift Service on AWS. The content in this guide is modular: you can choose to read the chapters in order or simply search for the information you need. If you're reading this e-book digitally, you can use the navigation buttons at the top of the page to jump between chapters and the table of contents.

If you have questions that aren't answered in this guide, please **contact us** so we can connect you with an account team focused on your success.

Who should read this e-book

This e-book is a guide for technical audiences—including developers and operations staff—who want to enhance their applications and processes with familiar AWS tools and services directly integrated with OpenShift Service on AWS.

What this e-book covers

This guide covers key topics for understanding and using AWS development tools with OpenShift Service on AWS. We begin with an introduction to Red Hat OpenShift and the reasons why many developers and operators choose it as their application platform, and how they derive value from it. Then we'll explain how you can integrate 2 key AWS services—Amazon Cognito and Amazon Cloudwatch—with OpenShift Service on AWS. Chapters 5 and 6 provide a hands-on guide for setting up both AWS services and OpenShift Service on AWS to implement the integration. Finally, in the last chapter, we offer helpful next steps.

Chapter 2

Introduction to Red Hat OpenShift

This chapter provides an introduction to Red Hat OpenShift, including what it is, how it is used, and the benefits it can deliver. Whether you're just getting started or you're already familiar with the platform, this chapter serves as a useful primer on the value and importance of Red Hat OpenShift.

Overview of Red Hat OpenShift

Red Hat OpenShift is a trusted, comprehensive, and consistent application platform for developing, deploying, modernizing, and managing traditional and cloud-native applications across hybrid cloud environments.

With support for a variety of environments, applications, and skill levels, Red Hat OpenShift lets developers build innovative applications in less time while boosting the performance of critical workloads. You can access validated images and solutions from hundreds of partners, with security scanning and cryptographic signing throughout the delivery process. Or use on-demand images and natively access a wide range of third-party cloud services, all through a single platform.

By streamlining essential IT activities on a unified application platform, Red Hat OpenShift helps operations teams increase efficiency and maintain security of complex hybrid environments. Gain visibility into deployments across multiple locations and teams with built-in logging and monitoring. Embed application logic for native and third-party services with Red Hat OpenShift operators to simplify configuration, performance tuning, update, and patching tasks.

Red Hat OpenShift helps organizations empower their IT operations and development teams. Many joint Red Hat and AWS customers choose Red Hat OpenShift Service on AWS (ROSA) as their preferred platform to speed application development and delivery across hybrid cloud environments.

Can't I just use Kubernetes?

Kubernetes is an important open source project—it is one of the Cloud Native Computing Foundation's key projects and an essential technology for running containers. As a result, you may wonder if you can simply run y our applications using Kubernetes alone.

Many organizations start by deploying Kubernetes and can get a container, or even a simple enterprise application, running in a few days. However, they increasingly find themselves building and managing their own application platform, based on Kubernetes technology, as they move into Day 2 operations, implement security requirements, and deploy more applications. For example, they might add an open source ingress controller, write scripts to connect their continuous integration/continuous deployment (CI/CD) pipelines, and try to deploy more complicated applications. At this point, the complexity of Day 2 management operations often becomes overwhelming.

Dealing with the complexity of building and maintaining a successful custom application platform usually takes an operations team of several people and weeks to months of effort. This can lead to inefficiency in the organization, complicated security and certification processes, and greater time and effort spent when onboarding developer teams.

The tasks involved with building, maintaining, and supporting a Kubernetes-based application platform can be grouped into 3 categories:

- **Cluster management**, including operating system installation and patching, Kubernetes installation, container network interface (CNI) configuration, authentication integration, ingress and egress setup, persistent storage setup, node hardening, security patching, and underlying cloud configuration.

- **Application services**, including log aggregation, health checks, performance monitoring, security patching, container registry management, and application staging process setup.

- **Developer integration**, including integration of CI/CD pipelines, developer tools, integrated development environments (IDEs), and frameworks; middleware compatibility testing; application performance dashboard setup; and role-based access control (RBAC) configuration.

This list is only a sample of the activities needed to start running containers in production. Even so, the time, effort, and understanding needed to complete these tasks is insignificant when compared to the ongoing maintenance of the platform and its individual components. Each component has its own release cycle, security policies, and patches and each integration must be thoroughly tested with each change and update.

Chapter 3

Introduction to Red Hat OpenShift Service on AWS

Red Hat OpenShift Service on AWS (ROSA) is a turnkey platform for developing, deploying, and administering applications across hybrid cloud environments. As a managed Red Hat OpenShift offering on AWS, OpenShift Service on AWS provides a complete application platform with fully integrated development and operational productivity features that let you build, operate, and scale globally and on demand through a familiar management interface. With IDEs, runtimes, build pipelines, monitoring tools, logging application programming interfaces (APIs), and service mesh capabilities—all built on a Kubernetes core—OpenShift Service on AWS is an ideal foundation for your containerized applications.

Key benefits of OpenShift Service on AWS

As a joint, cloud-native offering from Red Hat and AWS, OpenShift Service on AWS provides key benefits for organizations with cloud-native and hybrid cloud environments.

- **Streamlined integrations.** OpenShift Service on AWS is directly accessible from the AWS console, so you can simply integrate native AWS toolsets, applications, and services.
- **Expert support.** Red Hat and AWS jointly operate and support OpenShift Service on AWS with an integrated support experience and 99.95% uptime service-level agreement (SLA).
- **Comprehensive features.** OpenShift Service on AWS includes built-in tools and services as well as integrations with native AWS applications services like AWS App Runner, Elastic Load Balancing, AWS Directory Service, Amazon CloudWatch, and AWS X-Ray.
- **Simplified billing.** A single invoice from AWS for both your Red Hat OpenShift service and AWS infrastructure consumption streamlines and simplifies purchasing processes.
- **Cost-effective procurement.** Use AWS committed spend, negotiated discounts, and entitlements to purchase OpenShift Service on AWS.
- **Reliable operations.** OpenShift Service on AWS is backed by **expert site reliability engineers (SREs)** who automate the deployment and management of Red Hat OpenShift clusters.

Maximize your benefits with Red Hat OpenShift cloud services

Red Hat OpenShift cloud services—including OpenShift Service on AWS—build on the benefits of Red Hat OpenShift to deliver even more value. The Forrester study titled **The Total Economic Impact™ of Red Hat OpenShift Cloud Services** highlights several key financial benefits:

- 468% return on investment[1]
- US$4.08 million net present value (NPV)[1]
- 6-month payback time[1]

Beyond these financial benefits, the customers interviewed for the Forrester report experienced:

- **Faster development cycles.** Using Red Hat OpenShift cloud services allowed organizations to shorten their development cycle by up to 70%. By using an application platform with built-in tools—as well as the flexibility to use preferred cloud-native tools—organizations were able to spin up environments faster and focus on high-priority activities like responding to customer needs.[1]
- **More focused development teams.** Interviewees noted that Red Hat OpenShift cloud services eliminated the need for developers to maintain the application development infrastructure, allowing them to fully focus on building products and solutions. Over 3 years, this recaptured developer time was worth more than US$2.13 million.[1]
- **50% greater operational efficiency.** Interviewees noted that using these managed service solutions meant they could reassign 50% of DevOps employees who were previously responsible for managing the infrastructure to other work that is more productive.[1] Over 3 years, this increased operational efficiency was worth more than US$1.3 million.[1]

1 Forrester. "**The Total Economic Impact™ of Red Hat OpenShift Cloud Services**," February, 2024.

Migrate your virtual machines to a cloud-ready platform

For organizations looking to migrate and modernize virtual machine workloads, **Red Hat OpenShift Virtualization on OpenShift Service on AWS** offers a solution to run virtual machines and containers on a single enterprise software foundation. Included with your OpenShift Service on AWS subscription, Red Hat OpenShift Virtualization lets you create, import, clone, migrate, and manage Linux® and Microsoft Windows virtual machines on a modern application platform. In addition to running virtual machines, Red Hat OpenShift Virtualization on OpenShift Service on AWS provides:

- Integrated tools and capabilities to build, modernize, and deploy applications with both virtual machine- and container-based workloads.
- Consistent and cost-effective operation across hybrid and multicloud environments.
- Self-service provisioning options for deployment of virtual machines and integration with CI/CD pipelines.

Read the Red Hat OpenShift documentation to learn more about **Red Hat OpenShift Virtualization on OpenShift Service on AWS**.

Accelerate innovation with AI-enabled applications

Integrating artificial intelligence and machine learning (AI/ML) technologies into key applications can help organizations enhance customer experiences and increase their competitive advantage. **Red Hat OpenShift AI** uses the capabilities of OpenShift Service on AWS to provide a reliable AI/ML platform for building, training, tuning, deploying, and monitoring intelligent applications across hybrid cloud environments. With a choice of AI/ML technologies from a robust partner ecosystem—including **IBM watsonx**, **Amazon SageMaker**, **NVIDIA**, **Run:ai**, **Elastic**, and **Starburst**—Red Hat OpenShift AI lets you select the best tools for your business needs.

OpenShift Service on AWS architecture overview

OpenShift Service on AWS features a streamlined architecture, offering a robust Kubernetes foundation that enhances security and reliability for applications on AWS. The architecture supports both hybrid and cloud-native deployment patterns, empowering developers with familiar tools and reducing cluster management overhead. By integrating Red Hat OpenShift software components with AWS services, OpenShift Service on AWS provides a cohesive and efficient environment familiar to many Kubernetes users. These commonly used AWS services include:

- Amazon Elastic Compute Cloud (EC2)
- AWS Elastic Load Balancing (ELB)
- Amazon Elastic Block Store (EBS)
- Amazon S3
- AWS Virtual Private Cloud (VPC)
- Amazon Route 53
- AWS Security Token Service (STS)
- AWS Identity and Access Management (IAM)
- AWS PrivateLink
- AWS Key Management Service (KMS)

Users running self-managed Red Hat OpenShift on-site should note that in OpenShift Service on AWS, AWS services replace many functions that are normally provided by local services, including the domain name system (DNS), storage appliances, and infrastructure.

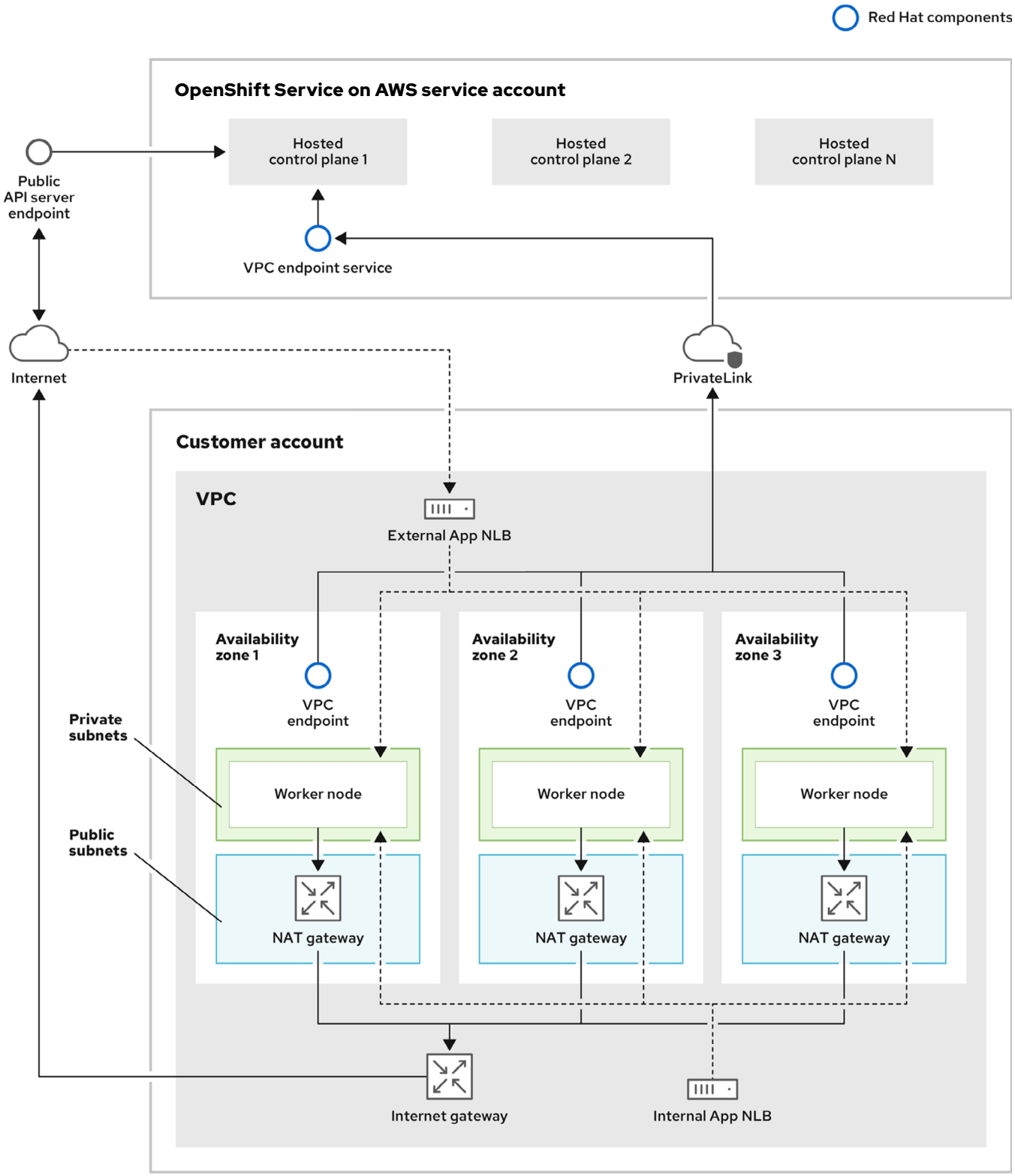


Figure 1. Network topology of an OpenShift Service on AWS cluster with hosted control planes

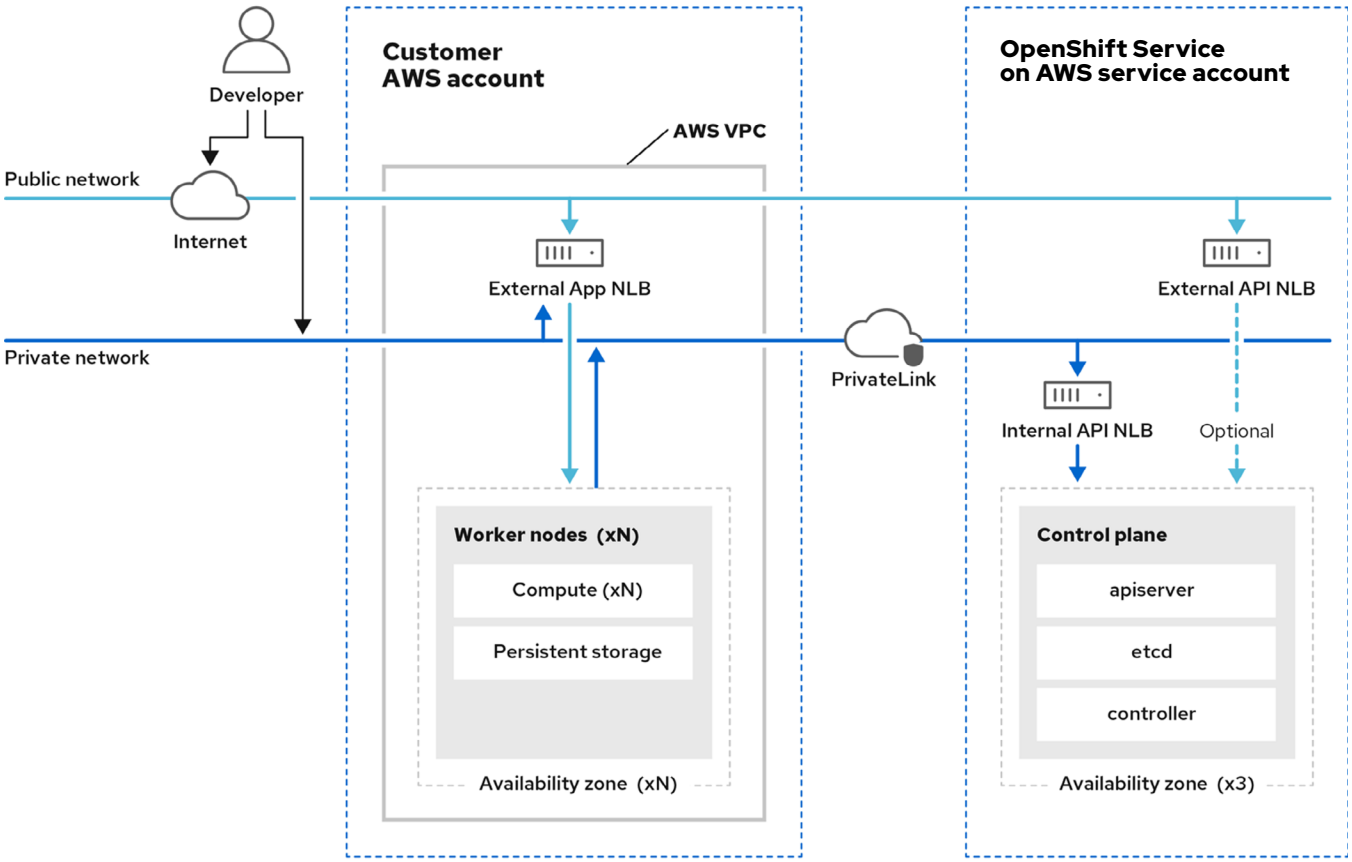


Figure 2. An OpenShift Service on AWS cluster with hosted control places in a private API network deployment

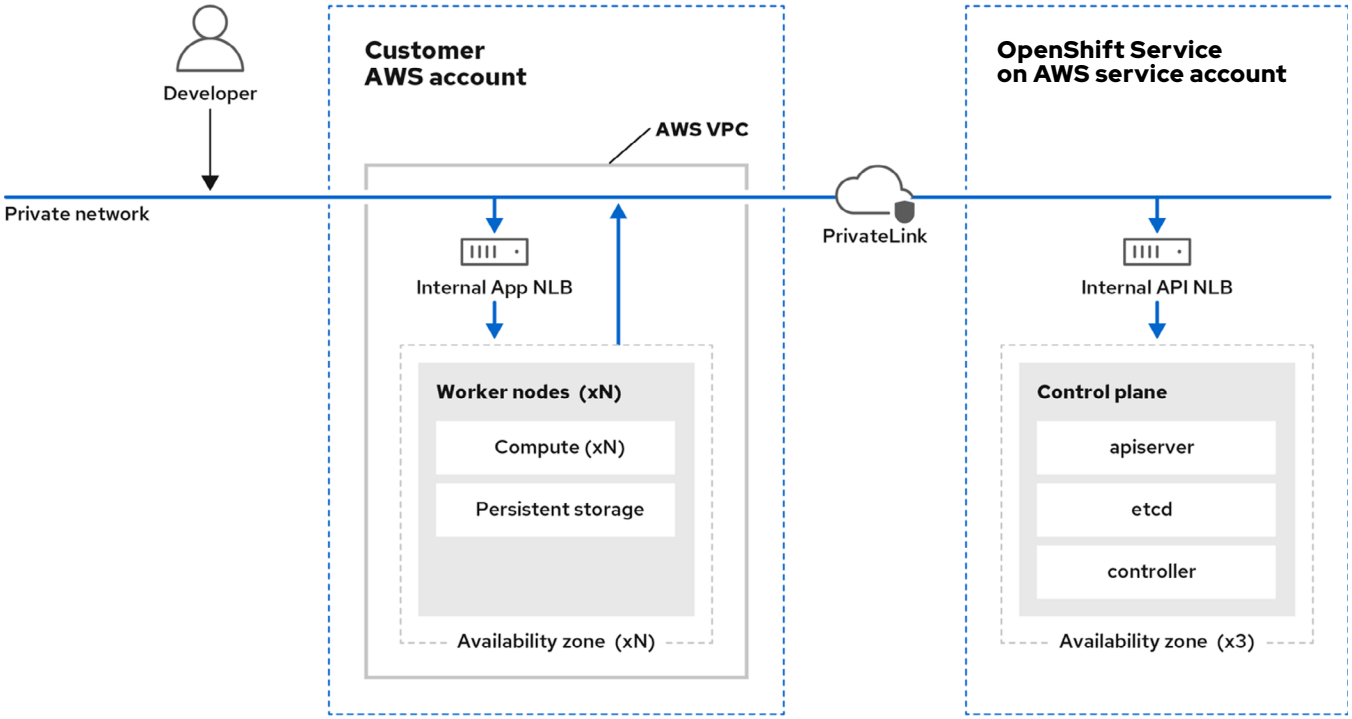


Figure 3. An OpenShift Service on AWS cluster with hosted control planes in a public API network deployment

Foundation

Red Hat OpenShift is based on **Red Hat Enterprise Linux® CoreOS**, an immutable operating system designed to run containerized software. Red Hat Enterprise Linux CoreOS hosts the essential software components for each cluster node's functionality. Each virtual machine instance in a cluster runs the kubelet service, a core component of Kubernetes architecture that manages container pods on each node according to instructions from the control plane. The state of all elements in Red Hat OpenShift is stored in etcd—a reliable, clustered key/value store that serves as the authoritative source of truth for the cluster. Refer to the OpenShift Service on AWS documentation to learn more about its **foundational architecture**.

Application compute

OpenShift Service on AWS clusters contain worker nodes—also known as compute nodes—that run your applications and workloads. Machine pools are logical groups of worker nodes that are allocated to specific AWS availability zones (AZs), sharing the same configuration and fault domains. When you build an OpenShift Service on AWS cluster, it initially includes at least 2 worker nodes that are part of the cluster's first machine pool.

Clusters can have many machine pools. By configuring several machine pools—each within a different AZ—you can distribute application pods across nodes with multiple replicas for redundancy. Each machine pool has a distinct configuration, including the Amazon EC2 instance type and the associated Amazon EBS volumes that store the root operating systems for the nodes.

Additionally, each machine pool in an OpenShift Service on AWS cluster can run a different version of Red Hat OpenShift, providing flexibility in managing upgrades. The cluster's control plane operates within the OpenShift Service on AWS management plane, and you can select the version for your cluster. The version of the control plane must be the same as or newer than the versions of the machine pools. In other words, machine pools cannot have a newer version than the cluster's control plane. Red Hat and AWS fully support your cluster as long as both the control plane and machine pools are within the support lifecycle.

Support and management

Beyond version updates, the management and health of your cluster's control plane are fully handled by Red Hat and AWS. The control plane automatically scales to match the size of your cluster, while the request-serving components are highly available and distributed within the same region where your cluster was created.

Networking

You must satisfy a basic set of network requirements to establish a fully functional OpenShift Service on AWS cluster. This includes a minimum number of subnets within your Amazon VPC to support both private and public EC2 instance networking, as well as AWS elastic load balancers. You need 1 network load balancer to serve as the default ingress (or router) for directing traffic to your applications, and another load balancer for managing traffic to your cluster's Kubernetes API. Optionally, you can set up additional load balancers for specific services advertised by your applications, allowing for customized traffic management.

Within your OpenShift Service on AWS cluster, the service network, also known as the container network interface, is managed by OVN-Kubernetes. This overlay network uses Geneve network virtualization encapsulation to connect all nodes, pods, and services in the cluster. Each node runs Open vSwitch, supporting declarative network operations controlled by OVN-Kubernetes. As a vendor-agnostic networking solution supported by Red Hat, OVN-Kubernetes provides features like ingress and egress rules and network policies.

Red Hat OpenShift operators

OpenShift Service on AWS includes Red Hat OpenShift operators that automatically manage your cluster. These operators consume a small amount of resources in your cluster. Some examples are:

- OpenShift Service on AWS Control Plane Operator
- OpenShift Service on AWS Ingress Operator
- OpenShift Service on AWS EBS Container Storage Interface (CSI) Driver Operator
- OpenShift Service on AWS Cloud Network Config Operator
- OpenShift Service on AWS Image Registry Operator

Managed control plane

The worker nodes in your cluster receive scheduling and operational commands from the control plane. This connection is established through a VPC endpoint in your AWS account, which allows worker nodes to connect to the OpenShift Service on AWS control plane via AWS PrivateLink. This setup ensures that all communication between your account and the OpenShift Service on AWS service is encrypted. Additionally, all communication within your cluster is securely encrypted.

Data plane

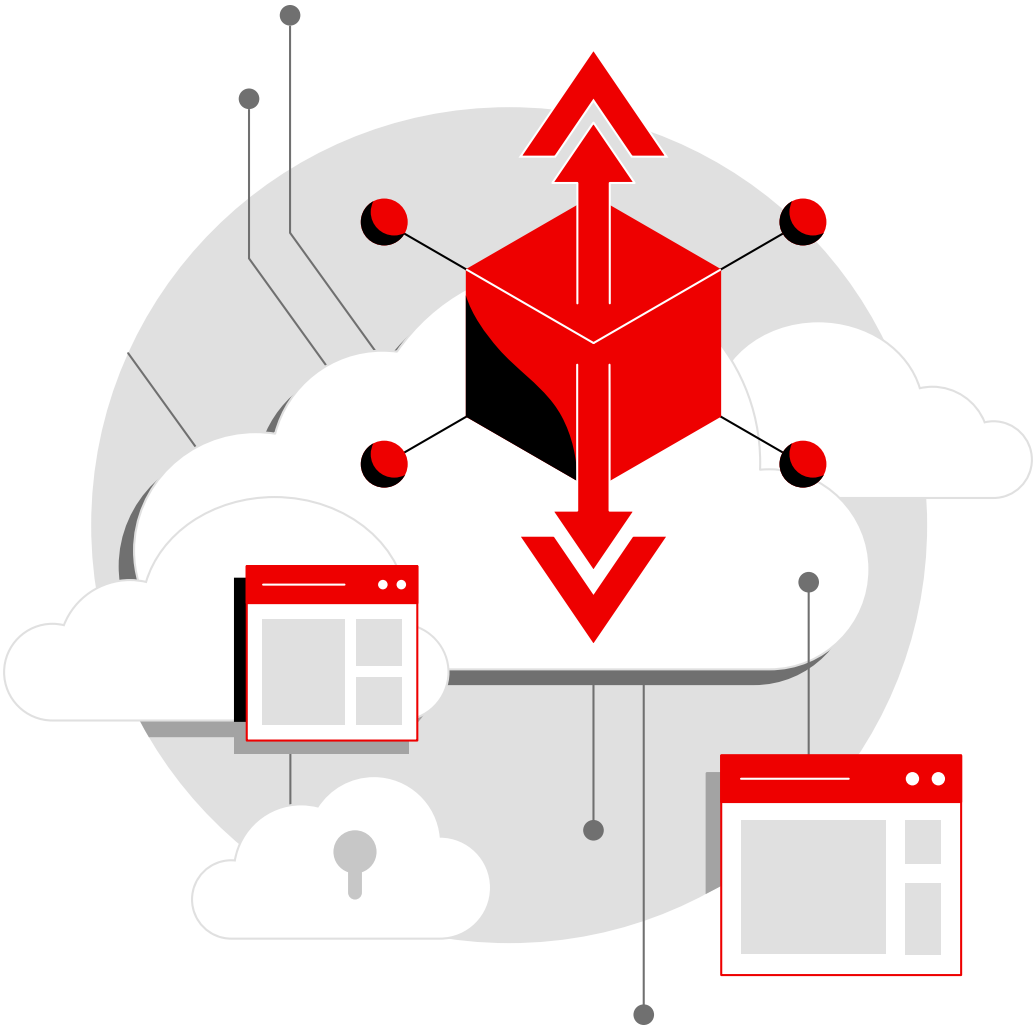
Storage for all persistent data in your cluster is encrypted at rest, following AWS default encryption practices. OpenShift Service on AWS clusters use the default account key in AWS KMS for encryption, unless you specify an alternative. This encryption applies to storage at rest for cluster nodes in your AWS account and the initial StorageClass for persistent volumes used by your workloads. Additionally, you can encrypt the etcd database for your OpenShift Service on AWS cluster using a specific AWS KMS key or the same key used for initial storage encryption.

Access control

As the owner of an OpenShift Service on AWS cluster, you can configure RBAC—including cluster-admin privileged access—to manage user permissions. Red Hat OpenShift provides RBAC levels that you can customize to meet the needs of your organization. This RBAC layer is separate from the AWS IAM controls governing your AWS account.

OpenShift Service on AWS clusters communicate with the AWS cloud API to automatically manage resources using AWS IAM roles. AWS STS provides the required temporary credentials using roles and policies defined during installation. All authentication and authorization for cluster components rely on these AWS IAM roles and managed policies for short-lived tokens for AWS API access.

The policies are continuously updated to ease management during cluster upgrades and changes in platform versions. AWS managed policies for OpenShift Service on AWS are carefully reviewed and approved by Amazon to ensure adherence to the principle of least privilege. Policies are also restricted by conditions based on resource tags, allowing the platform to perform actions only on resources with specific tags. This approach emphasizes the importance of maintaining a strong security posture and implementing secure cloud practices.



Chapter 4

Integrate OpenShift Service on AWS with AWS services

This chapter describes how to efficiently integrate key features between your Red Hat OpenShift Service on AWS (ROSA) environment and AWS services. This guide provides 2 example integrations that reflect best practices from cloud services experts at Red Hat and AWS. Each example outlines the steps to connect your OpenShift Service on AWS environment and AWS services and highlights the scalability, flexibility, and performance benefits of the integration.

This guide reviews the integration process using both the web console user interface (UI) and the command line interface (CLI). Some screen captures and commands may differ due to ongoing software updates. Review the steps carefully.

Because they are designed to simplify the integration process, these examples are not a substitute for official documentation or support. They should not be used to implement production-grade systems. You may need to review and adapt the steps to fit your specific needs and environments.

Set up environments and access

Before starting, ensure you have the following in place:

- **OpenShift Service on AWS installation.** This guide does not include set-up steps. You can install OpenShift Service on AWS from your Amazon account.
- **Administrative access.** You will need access to your AWS environment running OpenShift Service on AWS, Amazon Cognito, and Amazon CloudWatch. For production deployments, customize access according to your requirements.
- **Red Hat account.** Create a free account in the Red Hat Hybrid Cloud Console to access the Red Hat OpenShift Cluster Manager (OCM) environment used in these examples.

Using the CLI

The CLI-based examples require a CLI configured with the AWS and OpenShift Service on AWS CLI tools and the Red Hat OpenShift **oc** tools.

The AWS CLI is authenticated with your AWS account and configured for the region where OpenShift Service on AWS is running. Consult the **AWS CLI documentation** for more details.

The OpenShift Service on AWS CLI is authenticated with your Red Hat OpenShift Cluster Manager login. Use the following commands to confirm that the oc command is accessible to the OpenShift Service on AWS CLI.

```
rosa login
rosa list clusters
rosa verify openshift-client
```

Consult the **OpenShift Service on AWS getting started guide** for more details.

Using the web console UI

Use your web browser to follow the UI-based examples. You can find your clusters in **Red Hat OpenShift Cluster Manager**. To log in to the OpenShift Service on AWS console from Red Hat OpenShift Cluster Manager, select the ***Overview*** tab, then click on the ***Open console*** button in the upper right corner.

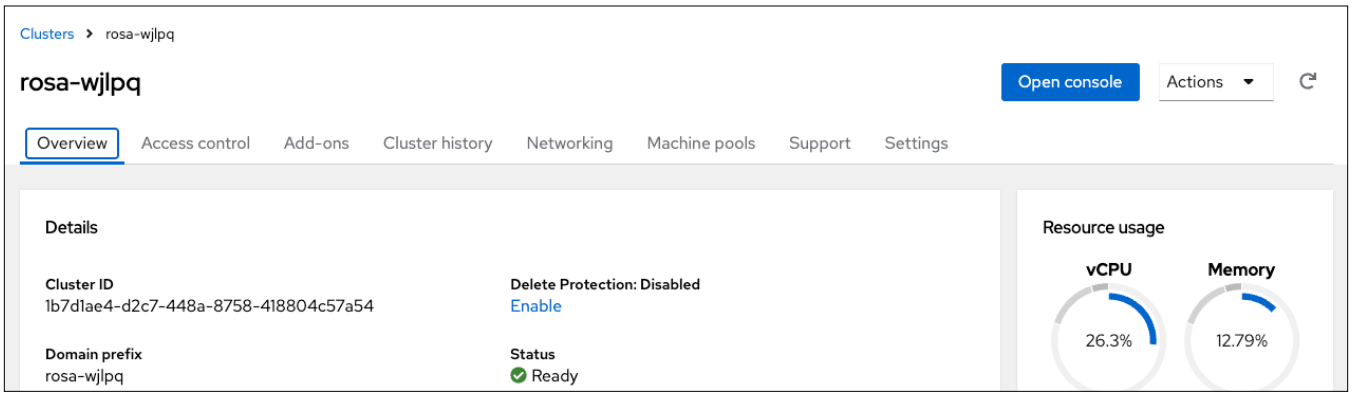


Figure 4. Log in to the OpenShift Service on AWS console from Red Hat OpenShift Cluster Manager.

Create a local admin user

Set up a local admin user for OpenShift Service on AWS using the htpasswd identity provider (IDP). Although this is not recommended for production environments, it is helpful for understanding the examples. There are 2 methods to create this user; these are briefly outlined below. For detailed instructions, refer to the **OpenShift Service on AWS getting started guide**.

Follow these steps to create a local admin user through the UI.

1. Navigate to the ***Access control*** section of your cluster in Red Hat OpenShift Cluster Manager.
2. Under ***Identity Providers***, select ***Add identity provider*** and ***htpasswd***.
3. Name your provider ***cluster-admin***.
4. Create a user called ***cluster-admin*** and set a password.
5. Under ***Cluster Roles and Access***, select ***Add user*** and enter the ***User ID*** as ***rosa-admin***.
6. Select ***cluster-admins*** as the group and click ***Add user***.

Follow these steps to create a local admin user through the CLI.

```
rosa create admin --cluster <rosa-cluster-name>
```

This command creates a simple IDP for OpenShift Service on AWS called ***rosa-admin*** and a ***cluster-admin*** user with full administrative access. It also generates a password and provides the following **oc** login command line:

```
oc login https://api.<rosa-cluster-name>.domain.p1.openshiftapps.com:6443 \
  --username cluster-admin --password <auto-generated-password>
```

Chapter 5

Centralize identity and access management with Amazon Cognito

Amazon Cognito is an AWS service that offers security-focused, scalable customer identity and access management, capable of supporting millions of users and devices. Operating on a fully managed, high-performance, reliable back end, this service lets you efficiently add enterprise-level user management to applications.

Red Hat OpenShift Service on AWS (ROSA) supports multiple authentication providers, while Amazon Cognito integrates with external IDPs that support SAML or **OpenID** Connect (OIDC). An interoperable authentication protocol based on the OAuth 2.0 framework of specifications, OIDC simplifies identity verification of users based on authentication performed by an authorization server. The examples in this chapter use the built-in OIDC IDP in OpenShift Service on AWS to connect with Amazon Cognito.

This chapter provides integration steps for both the CLI and the web console UI:

- Jump to **CLI instructions**.
- Jump to **Web console UI instructions**.

Integrate OpenShift Service on AWS and Amazon Cognito via the CLI

This section shows you how to use the CLI to integrate OpenShift Service on AWS and Amazon Cognito.

Create an Amazon Cognito user pool

Amazon Cognito user pools are OIDC IDPs that let you manage authorization and authentication through API calls. User pools provide an administrator-managed user directory that adds security, application integration, and customization features. In this example, all cluster users are entirely within AWS.

Create the Amazon Cognito user pool using the **`aws cognito-idp create-user-pool`** command. This command creates a user pool called ***rosa-pool*** that does not require email verification. Only administrators may create user configurations in this pool. All other settings use the Amazon Cognito defaults as seen in the command output.

```
aws cognito-idp create-user-pool --pool-name <rosa-pool> \
  --auto-verified-attributes email \
  --admin-create-user-config='{"AllowAdminCreateUserOnly": true}'
```

Create an Amazon Cognito user domain

Follow these steps to create a unique domain for your user pool. You can use an externally hosted domain with your own DNS or an Amazon Cognito hosted domain. This example uses an Amazon Cognito hosted domain.

1. Retrieve your user pool ID and export it as an environment variable.

```
export AWS_USER_POOL_ID=$(aws cognito-idp list-user-pools \
  --max-results 1 | jq -r .UserPools[0].Id)
```

2. Ensure the environment variable is set correctly. Find the pool ID in the Id: line of the output.

```
echo $AWS_USER_POOL_ID
```

3. Create an Amazon Cognito user domain. Specify the domain name and pool ID. This command does not return any output.

```
aws cognito-idp create-user-pool-domain --domain <rosa-domain> \
  --user-pool-id $AWS_USER_POOL_ID
```

Create users in Amazon Cognito

Follow these steps to create an administrative user and 2 regular users. Initially, all users are identical. Later steps use OpenShift Service on AWS to enable permissions and elevate privileges for the administrative user. Additionally, this example uses the **admin-create-user** subcommand for all users. It sets basic attributes and temporary passwords for all users. The Amazon Cognito user pool policy requires users to change them at first login.

1. Create the administrative user. Supply the pool ID using the exported environment variable.

```
aws cognito-idp admin-create-user \
  --user-pool-id $AWS_USER_POOL_ID \
  --username admin \
  --temporary-password <temp-password> \
  --user-attributes Name=name,Value="Cluster Administrator" \
                    Name="email",Value="admin@example.com" \
                    Name="email_verified",Value="true" \
  --message-action SUPPRESS
```

2. Create the regular users.

```
aws cognito-idp admin-create-user \
  --user-pool-id $AWS_USER_POOL_ID \
  --username user1 \
  --temporary-password <temp-password> \
  --user-attributes Name=name,Value="Cluster User1" \
                    Name="email",Value="user1@example.com" \
                    Name="email_verified",Value="true" \
  --message-action SUPPRESS
aws cognito-idp admin-create-user \
  --user-pool-id $AWS_USER_POOL_ID \
  --username user2 \
  --temporary-password <temp-password> \
  --user-attributes Name=name,Value="Cluster User2" \
                    Name="email",Value="user2@example.com" \
                    Name="email_verified",Value="true" \
  --message-action SUPPRESS
```

Create an Amazon Cognito user pool app client

App clients let applications interact with user pools. By connecting all components, app clients allow communication between OpenShift Service on AWS and Amazon Cognito. When an app client requires an authorization grant, it calls back to the location in the callback URL. This location resides on OpenShift Service on AWS and contains the oauth app, cluster, and domain names, along with the oauth2callback path with the IDP name. For this example, the IDP name is ***Cognito***. While the callback URL is cluster dependent, you can review the **`--callback-urls`** line to see the complete URL for this sample cluster.

Follow these steps to create an Amazon Cognito user pool app client.

1. Retrieve and store the callback URL.

```
CLUSTER_DOMAIN=$(rosa describe cluster -c <rosa-cluster-name> | \
  grep “DNS” | grep -oE ‘\S+.openshiftapps.com’)
```

2. Construct the callback URL.

```
echo “OAuth callback URL: \
  https://oauth-openshift.apps.${CLUSTER_DOMAIN}/oauth2callback/Cognito”
```

3. Create the Amazon Cognito user pool app client using the callback URL.

```
aws cognito-idp create-user-pool-client \
  --user-pool-id $AWS_USER_POOL_ID \
  --client-name <rosa-cluster-name> \
  --generate-secret \
  --supported-identity-providers COGNITO \
  --callback-urls \
“https://oauth-openshift.apps.${CLUSTER_DOMAIN}/oauth2callback/Cognito” \
  --allowed-o-auth-scopes “phone” “email” “openid” “profile” \
  --allowed-o-auth-flows code \
  --allowed-o-auth-flows-user-pool-client
```

4. Export the client ID and client secret to environment variables.

```
export AWS_USER_POOL_CLIENT_ID=$(aws cognito-idp list-user-pool-clients \
  --user-pool-id $AWS_USER_POOL_ID | jq -r .UserPoolClients[0].ClientId)
export AWS_USER_POOL_CLIENT_SECRET=$( \
  aws cognito-idp describe-user-pool-client --user-pool-id $AWS_USER_POOL_ID \
  --client-id ${AWS_USER_POOL_CLIENT_ID} | jq -r .UserPoolClient.ClientSecret)
```

5. Ensure the environment variables are set correctly.

```
echo -e "Client ID: ${AWS_USER_POOL_CLIENT_ID}\n \
         Client Secret: ${AWS_USER_POOL_CLIENT_SECRET}"
```

Create an OpenShift Service on AWS IDP

An IDP within OpenShift Service on AWS allows users to authenticate with Amazon Cognito. While OpenShift Service on AWS supports a variety of IDPs, this example uses an OIDC IDP.

Create an OpenShift Service on AWS IDP using the **`rosa create idp`** command.

```
rosa create idp \
  --cluster <rosa-cluster-name> \
  --type openid \
  --name Cognito \
  --client-id ${AWS_USER_POOL_CLIENT_ID} \
  --client-secret ${AWS_USER_POOL_CLIENT_SECRET} \
  --issuer-url https://cognito-idp.$(aws configure get \
               region).amazonaws.com/${AWS_USER_POOL_ID} \
  --email-claims email \
  --name-claims name \
  --username-claims username
```

The **`create idp`** subcommand lets you customize your OpenShift Service on AWS IDP. Here are some commonly used arguments. Consult the Managing objects with the OpenShift Service on AWS CLI documentation for further details.

- **`--cluster:`** Name of the OpenShift Service on AWS cluster containing the IDP
- **`--type:`** IDP type
- **`--name:`** Name of identity provider displayed by OpenShift Service on AWS
- **`--client-id:`** Amazon Cognito user pool client ID
- **`--client-secret:`** Amazon Cognito user pool client secret

- **`--issuer-url:`** URL that the OIDC IDP asserts as the issuer identifier using the https scheme with no URL query parameters or fragments
  Note: This example uses the Amazon Cognito domain with the AWS region and Amazon Cognito user pool ID.

- **`--<VAR>-claims:`** List of claims used for each type.

By filtering the **`oc get oauth`** command with **`jq`**, you can see a list of all available IDPs, including the cluster-admin and Amazon Cognito IDPs.

```
oc get oauth cluster -o json | jq -r '.spec.identityProviders[].name'
```

OpenShift Service on AWS may take up to 5 minutes to create the IDP. Rerun this command until it reports the new IDP.

Set up OpenShift Service on AWS permissions on Amazon Cognito users

OpenShift Service on AWS uses RBAC to assign permissions to user objects. Administrators bind collections of rules—called roles—to user objects to determine the actions users may perform. Roles are applied as policies and Red Hat OpenShift has a preconfigured administrative policy called ***cluster-admin***. This example uses that policy to grant permissions to the Amazon Cognito ***admin*** user. Consult the **Red Hat OpenShift RBAC documentation** for more details.

Add permissions for cluster administration to the Amazon Cognito ***admin*** user using the **`ocm adm policy`** command. This allows the ***admin*** user object managed by the Amazon Cognito IDP to be an OpenShift Service on AWS cluster administrator.

```
oc adm policy add-cluster-role-to-user cluster-admin admin
```

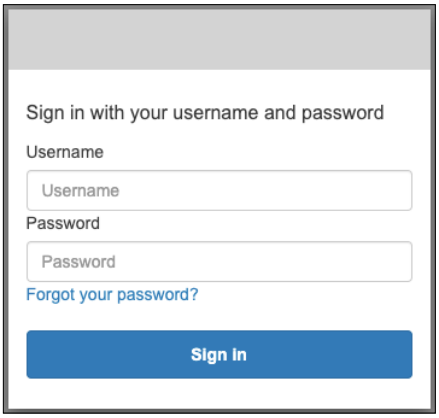
You may delete the local user from the htpasswd IDP or retain it for security purposes. In either case, ensure that both logins are well protected.

Log in to OpenShift Service on AWS using the Amazon Cognito IDP

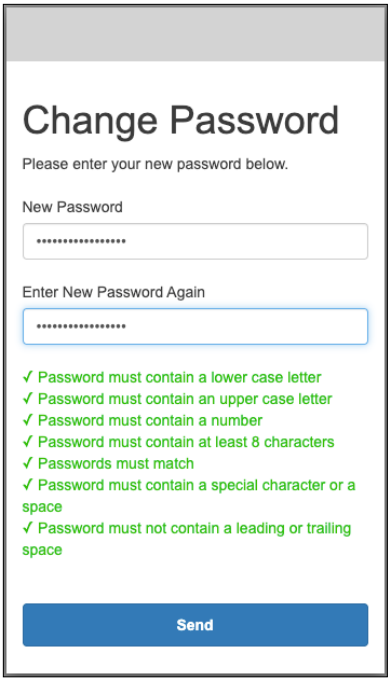
Follow these steps to log in and change your password.

1. Click ***Cognito*** on the OpenShift Service on AWS log-in screen to select the Amazon Cognito IDP.

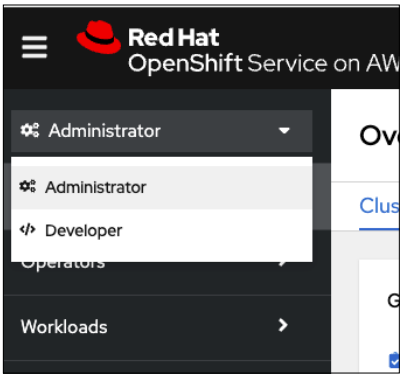
2. Provide your username and temporary password to authenticate with Amazon Cognito.



3. Change your password. Because this example uses **`--temporary-password`** when creating user accounts, Amazon Cognito requires all users to create a new password during their first login.



4. Confirm that the Amazon Cognito ***admin*** user can administer the cluster.



Integrate OpenShift Service on AWS and Amazon Cognito via the web console UI

This section shows you how to use the web console UI to integrate OpenShift Service on AWS and Amazon Cognito.

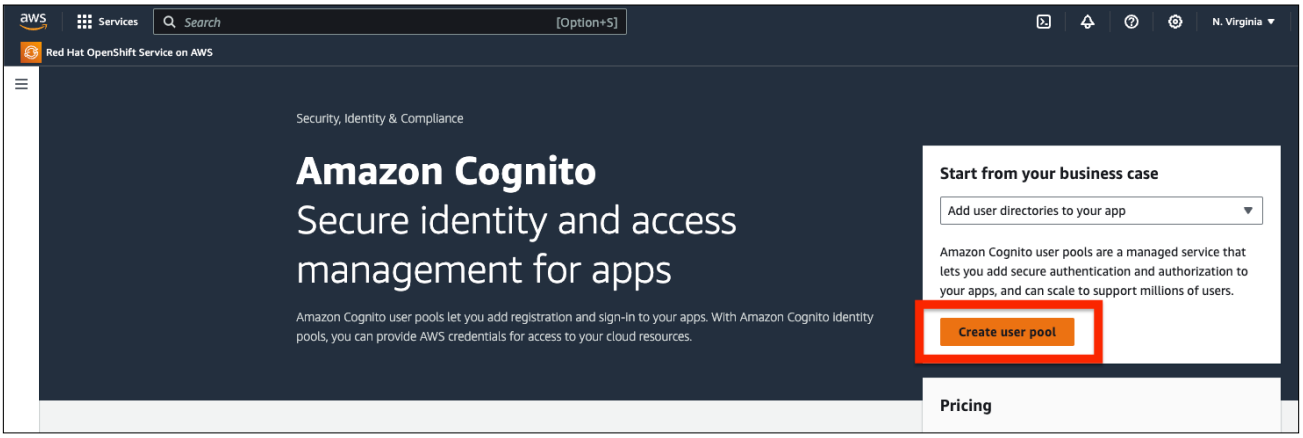
Create an Amazon Cognito user pool

Amazon Cognito user pools are OIDC IDPs that let you manage authorization and authentication through API calls. User pools provide an administrator-managed user directory that adds security, application integration, and customization features. In this example, all cluster users are entirely within AWS.

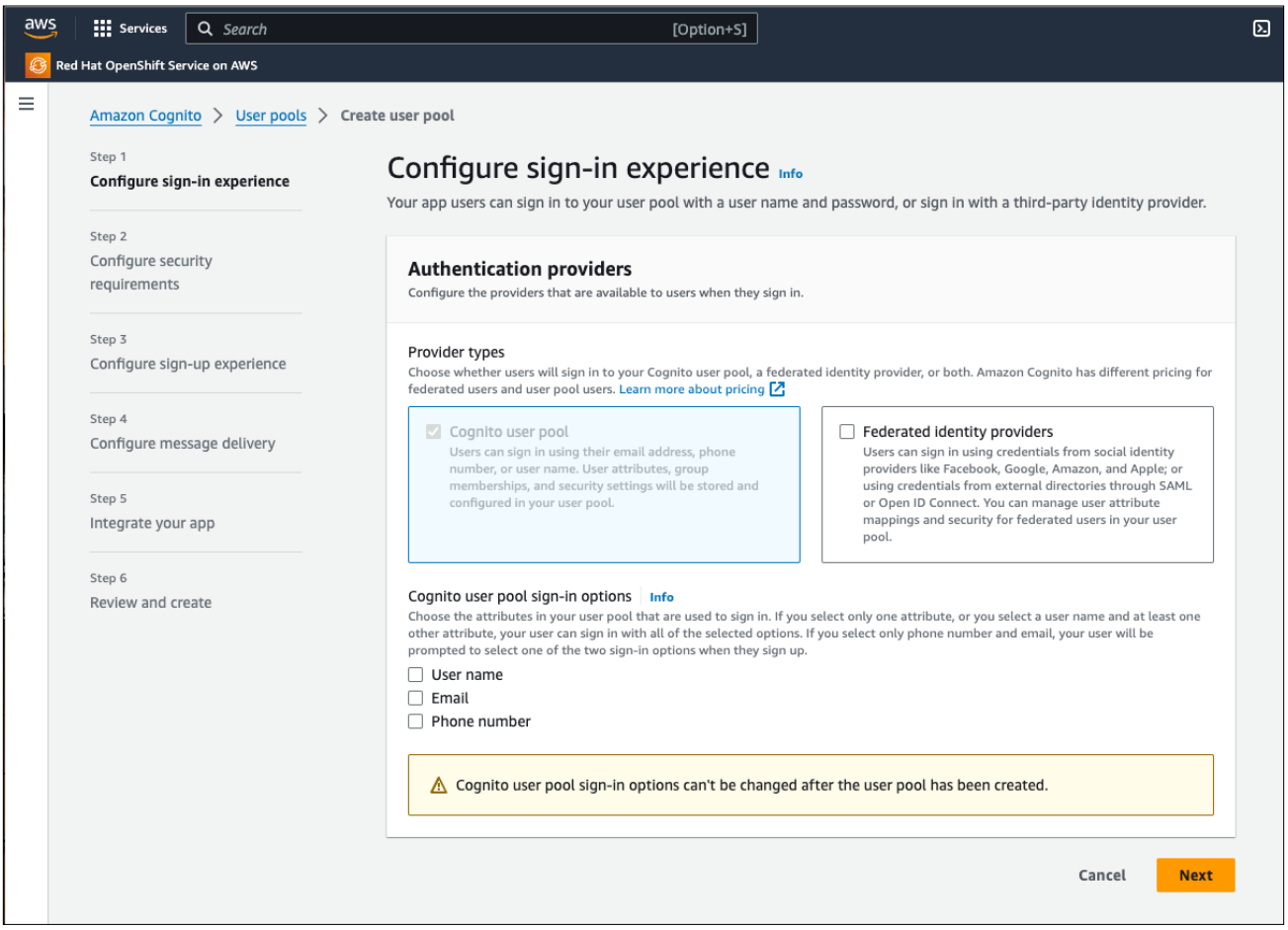
Follow these steps to create an Amazon Cognito user pool.

1. Log in to your AWS console and select the Amazon Cognito service for your region.

2. Click ***Create user pool***.



3. Complete the ***Configure sign-in experience*** form.

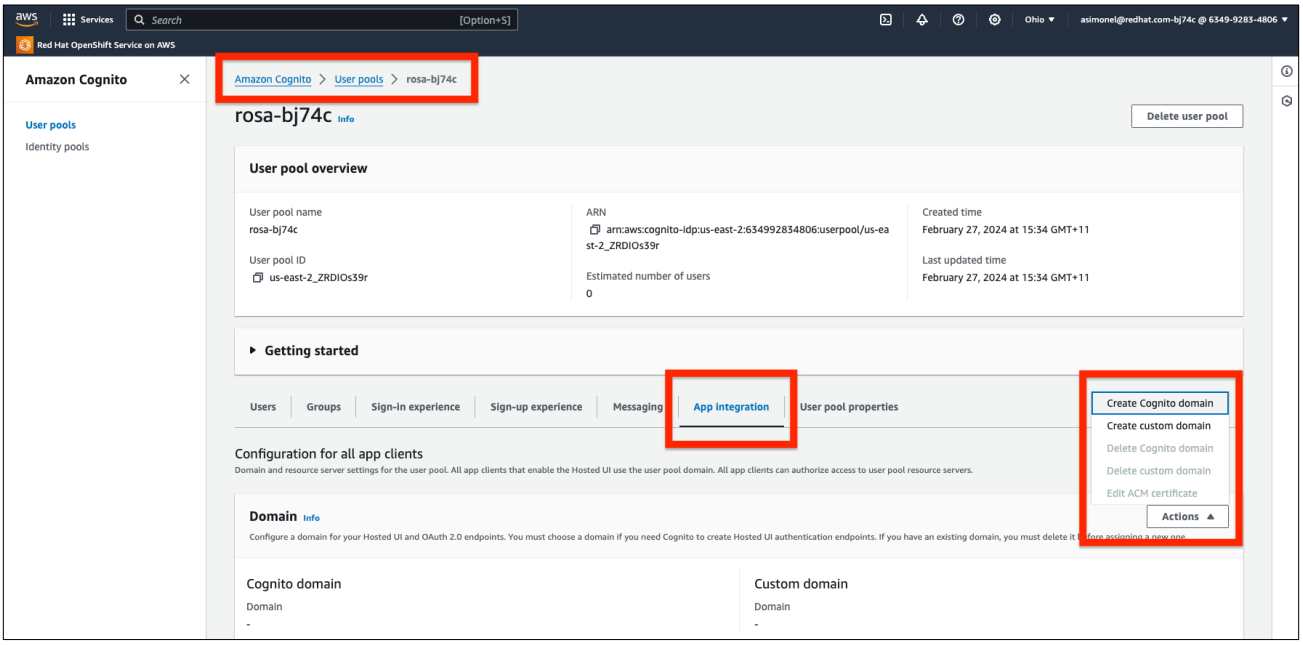


Create an Amazon Cognito user domain

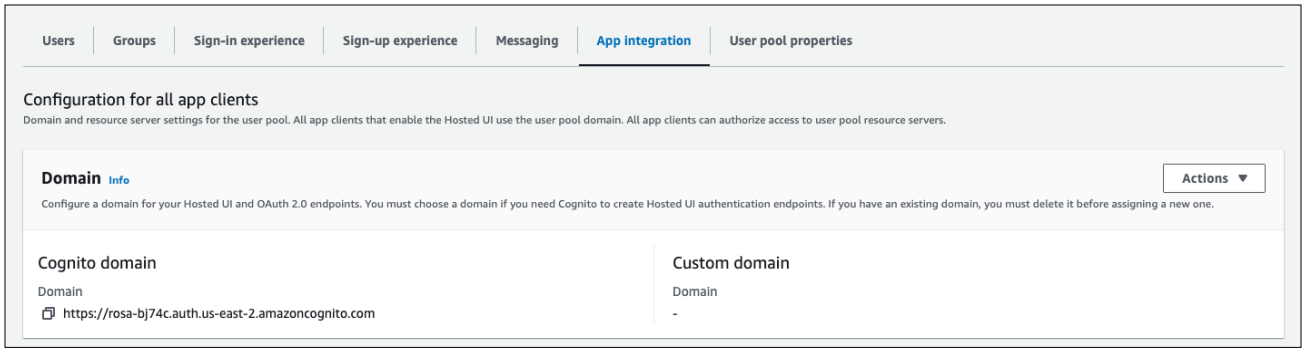
Follow these steps to create a unique domain for your user pool. You can use an externally hosted domain with your own DNS or an Amazon Cognito hosted domain. This example uses an Amazon Cognito hosted domain.

1. Navigate to the ***App integration*** tab in the new user pool.

2. Choose ***Create Cognito domain*** in the ***Actions*** menu.



3. View the status of the new domain in the ***App integration*** tab.

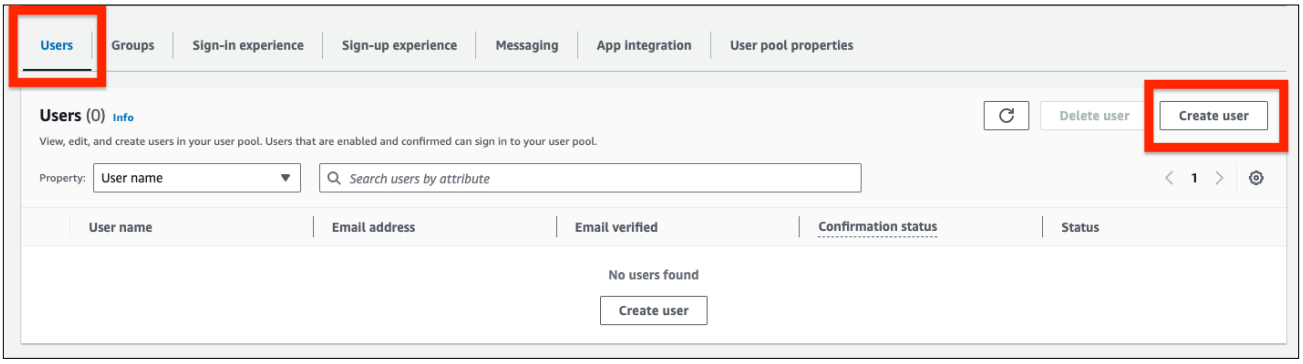


Create users in Amazon Cognito

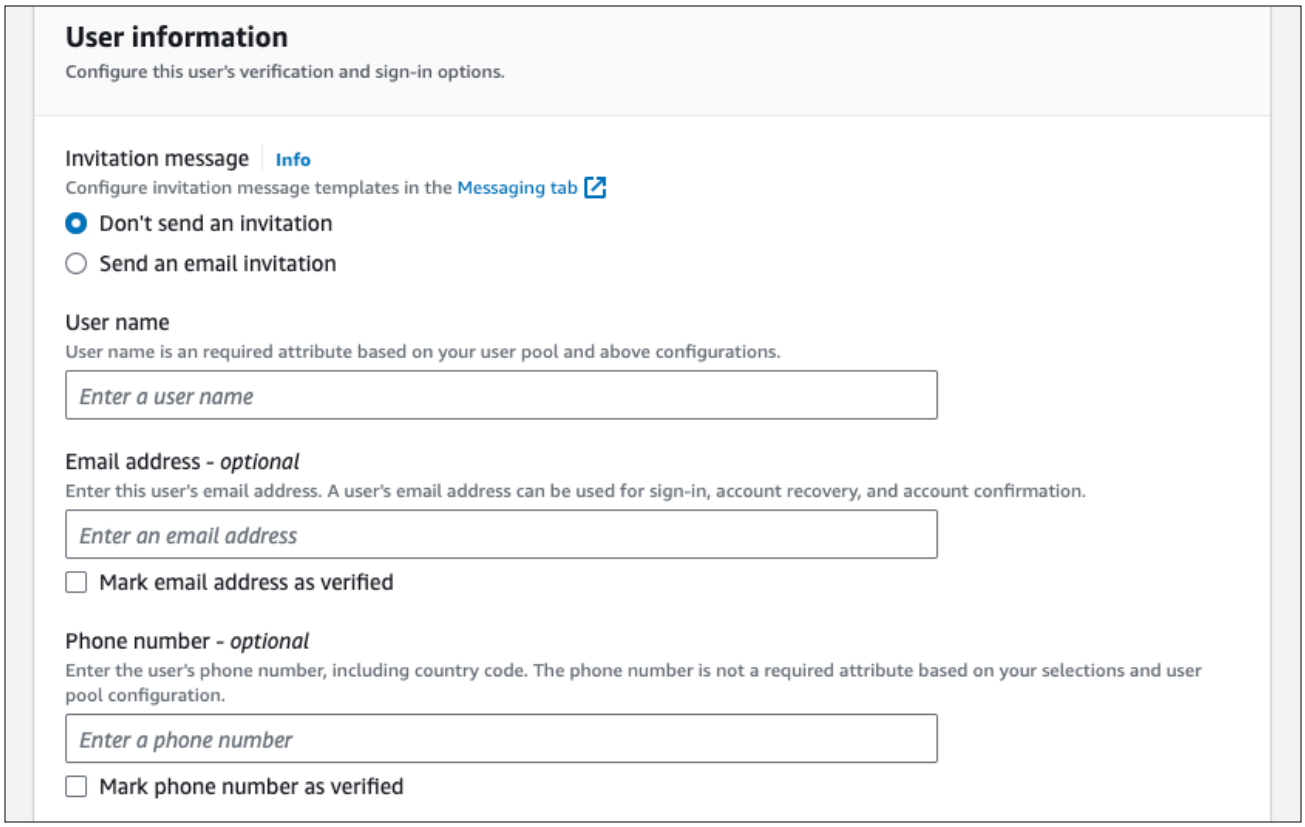
Follow these steps to create an administrative user and 2 regular users. Initially, all users are identical. Later steps use OpenShift Service on AWS to enable permissions and elevate privileges for the administrative user.

1. Navigate to the ***Users*** tab in the new user pool.

2. Click ***Create user***.

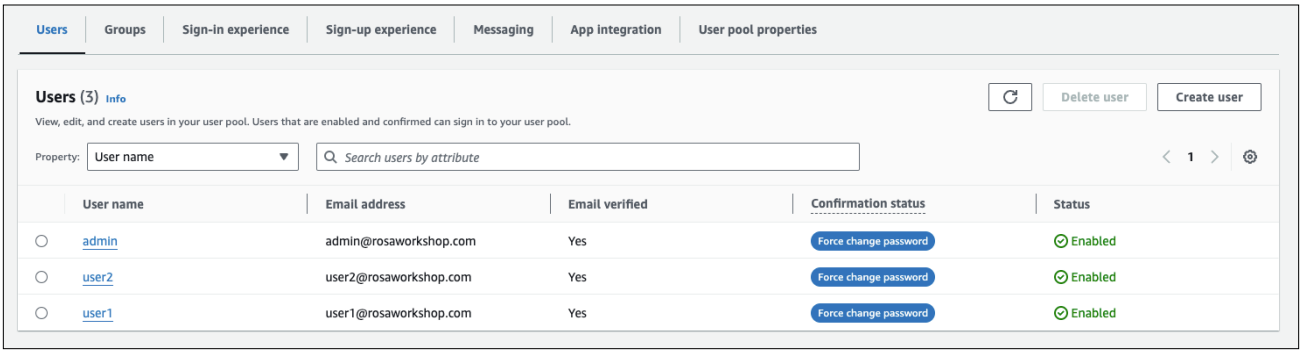


3. Complete the ***User information*** form.



4. Repeat this process for all three users: ***admin***, ***user1***, and ***user2***.

5. View the status of the new users in the ***Users*** tab.



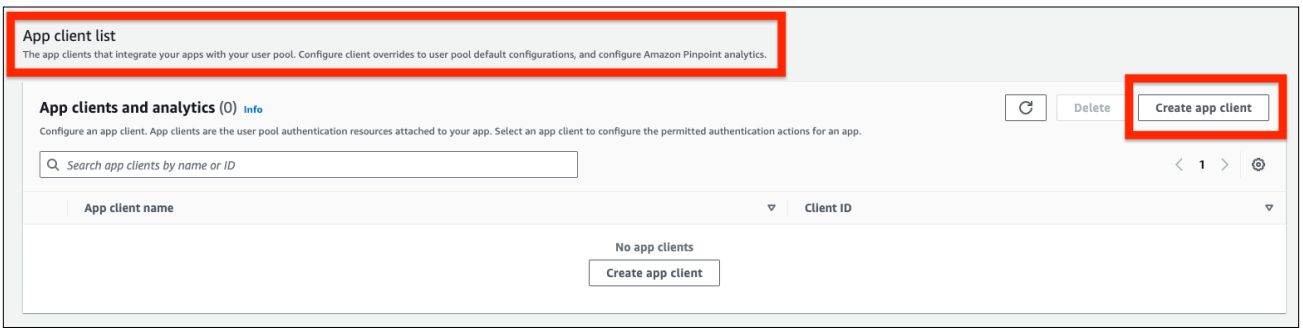
Create an Amazon Cognito user pool app client

App clients let applications interact with user pools. By connecting all components, app clients allow communication between OpenShift Service on AWS and Amazon Cognito. When an app client requires an authorization grant, it calls back to the location in the **callback URL**. This location resides on OpenShift Service on AWS and contains the oauth app, cluster, and domain names, along with the oauth2callback path with the IDP name. For this example, the IDP name is ***Cognito***. While the callback URL is cluster dependent, for this example, it is **https://oauth-openshift.apps.rosa-bj74c.ibhp.p1.openshiftapps.com/oauth2callback/Cognito**.

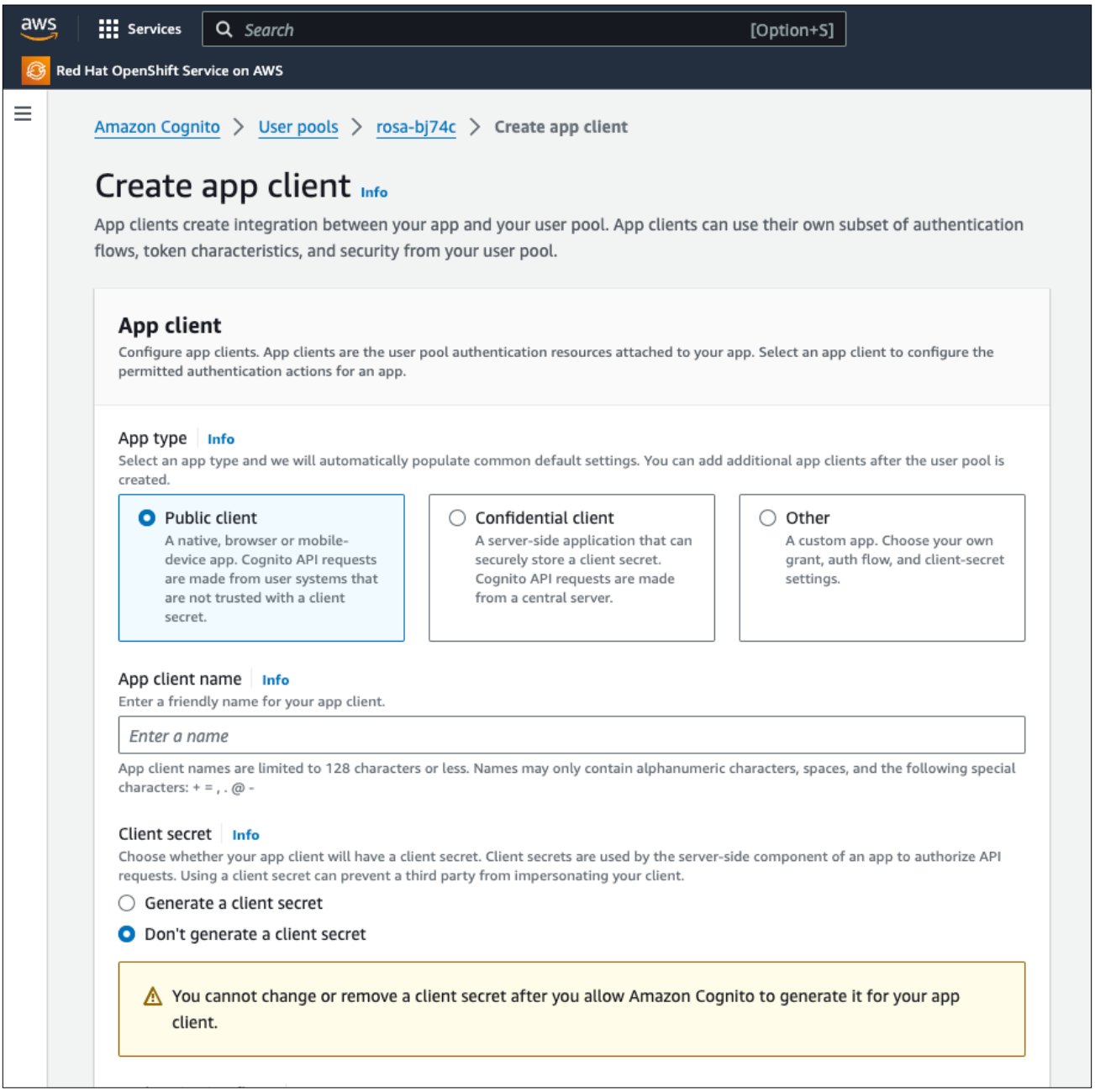
Follow these steps to create an Amazon Cognito user pool app client.

1. Navigate to the ***App client list*** on the ***App integration*** tab in the new user pool.

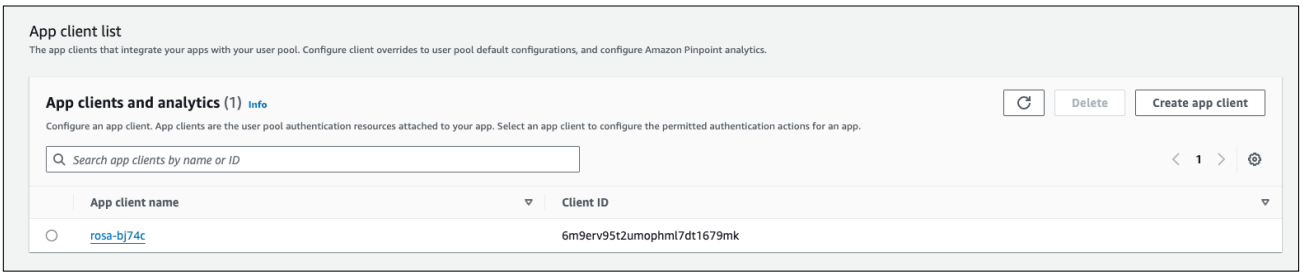
2. Click ***Create app client***.



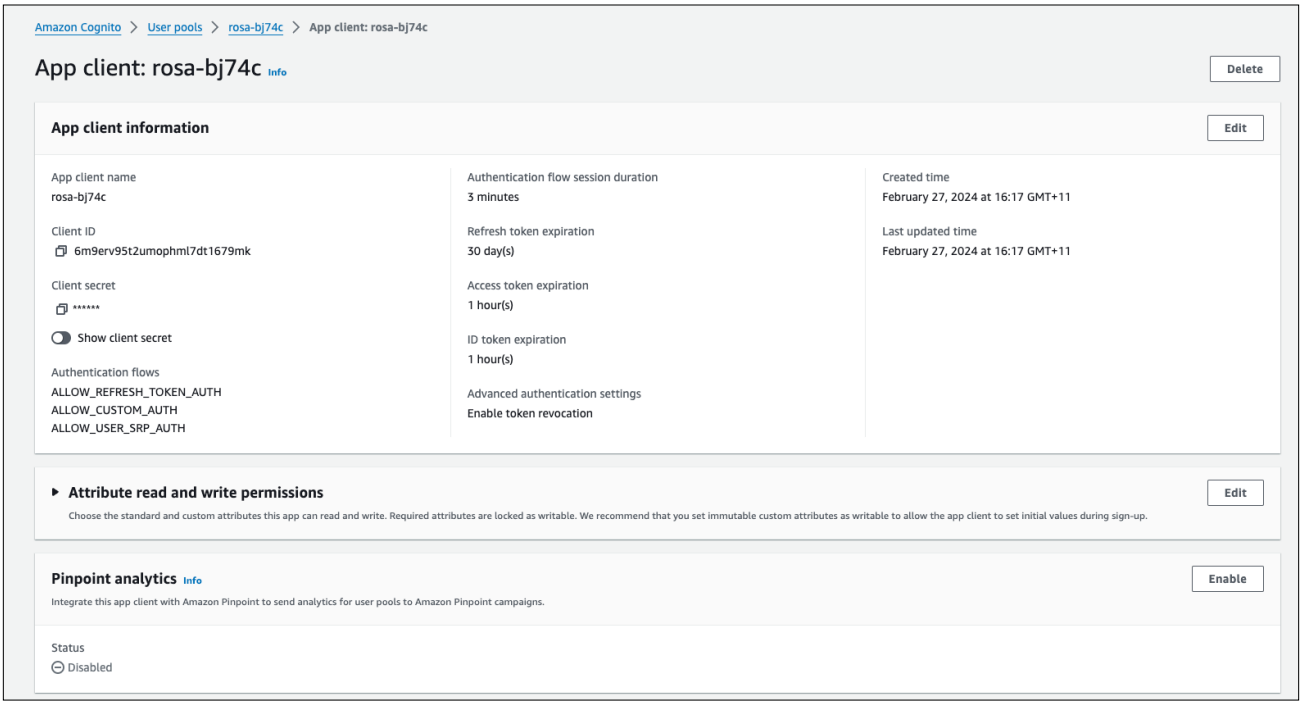
3. Complete the ***Create app client*** form.



4. Confirm that the new app client appears in the ***App client list***.



5. Select the new app client to view and edit details.

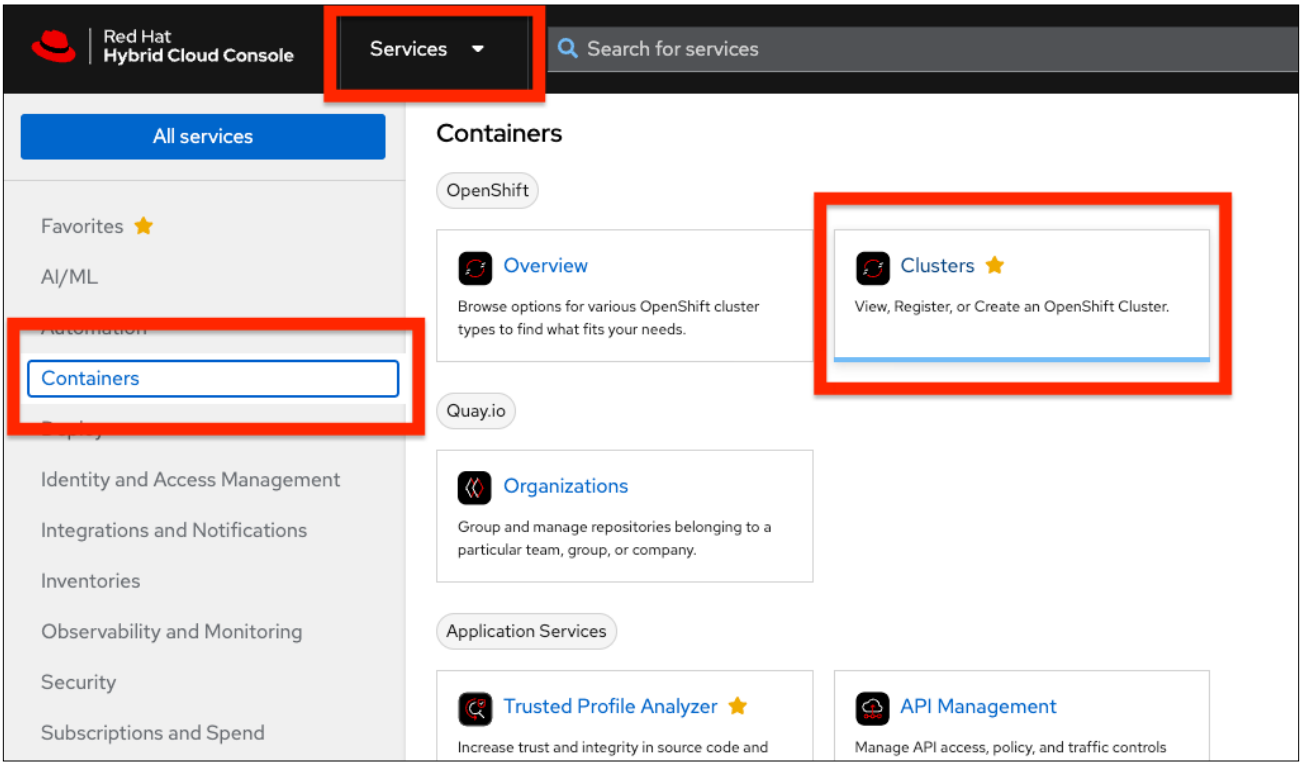


Create an OpenShift Service on AWS IDP

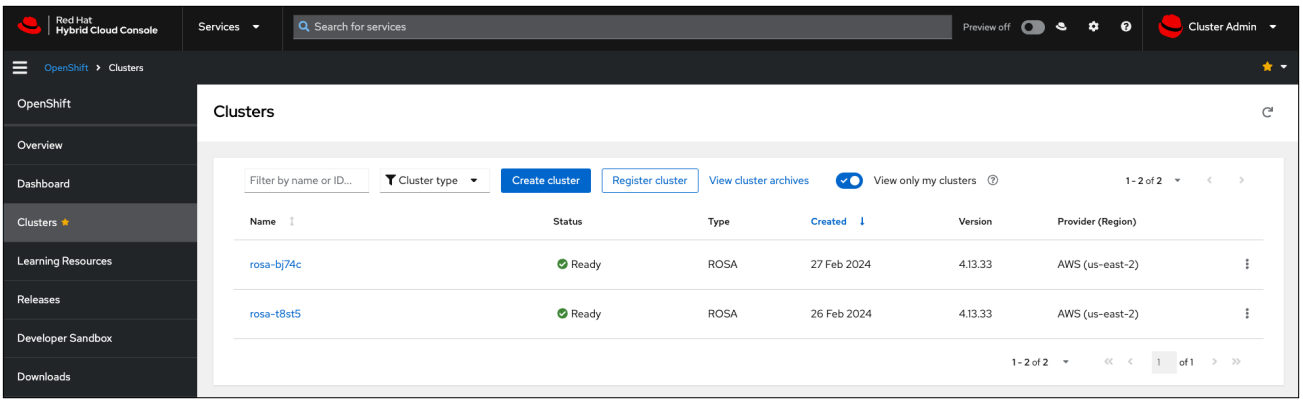
An IDP within OpenShift Service on AWS allows users to authenticate with Amazon Cognito. While OpenShift Service on AWS supports a variety of IDPs, this example uses an OIDC IDP.

Follow these steps to create an OpenShift Service on AWS IDP.

1. Navigate to ***Services*** > ***Containers*** > ***Clusters*** in Red Hat Hybrid Cloud Console.

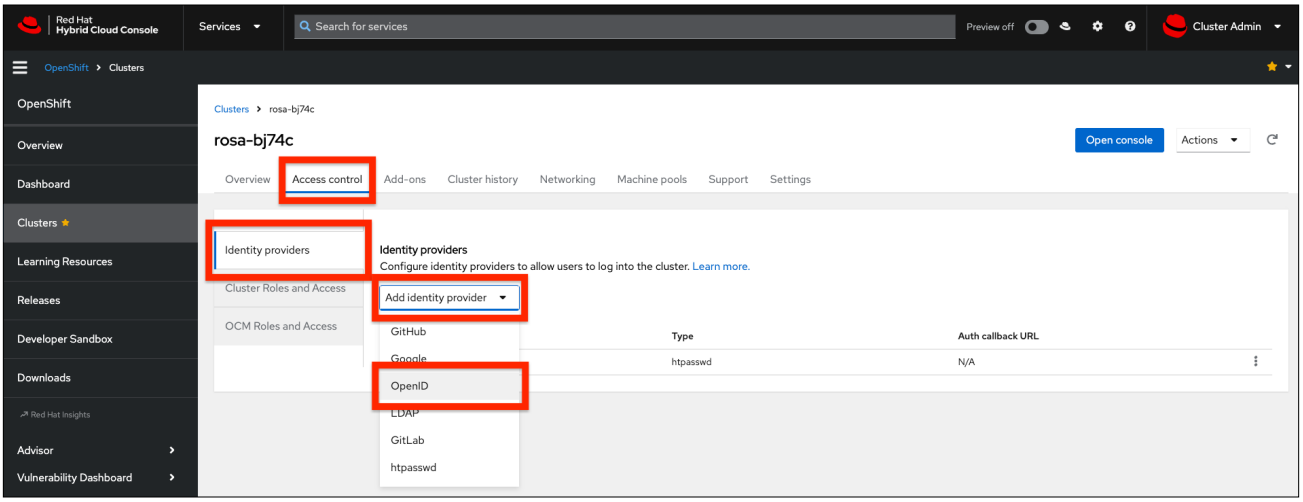


2. Select the cluster for the new IDP.

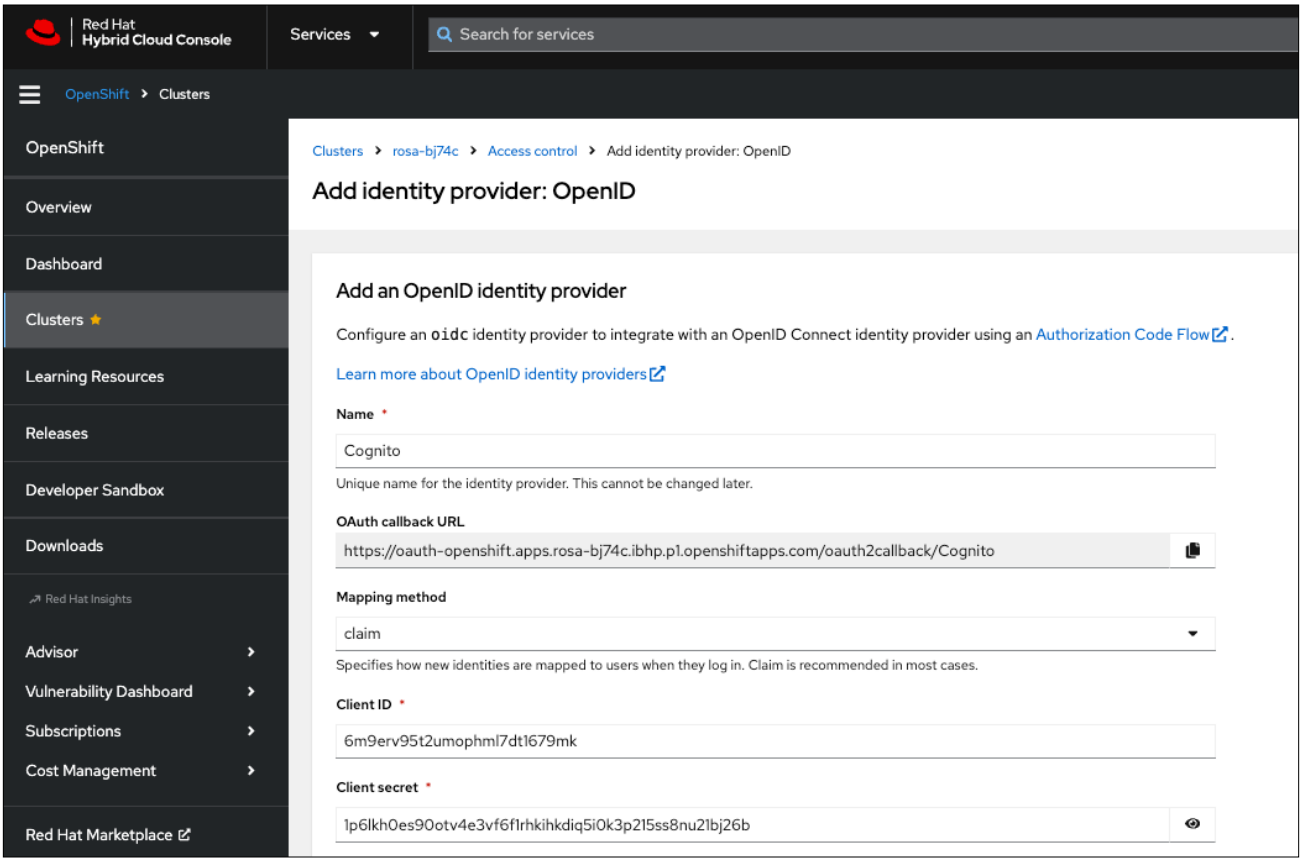


3. Navigate to ***Access Control > Identity providers***.

4. Choose ***OpenID*** in the ***Add identity provider*** menu.



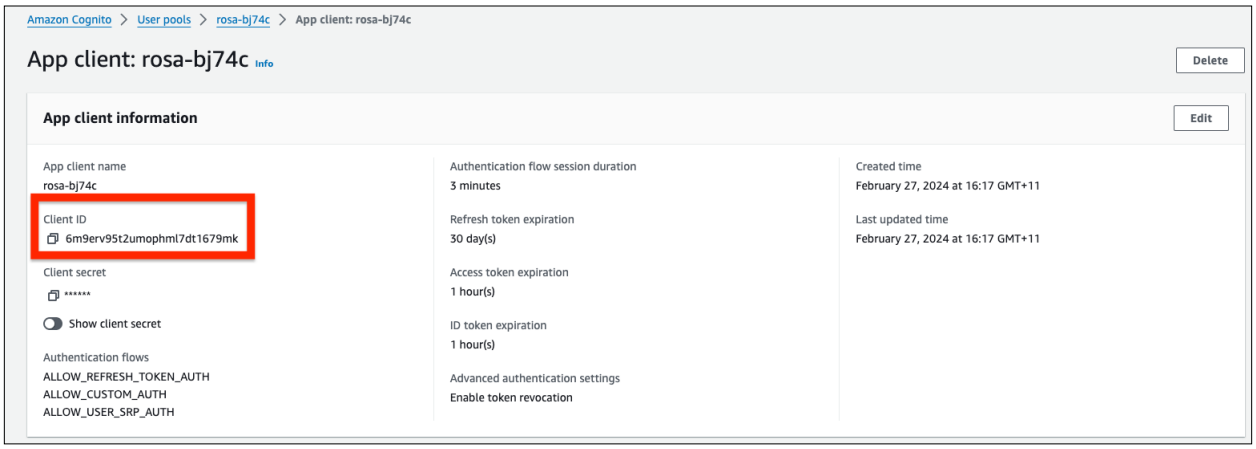
5. Complete the ***Add identity provider: OpenID*** form according to the following instructions.



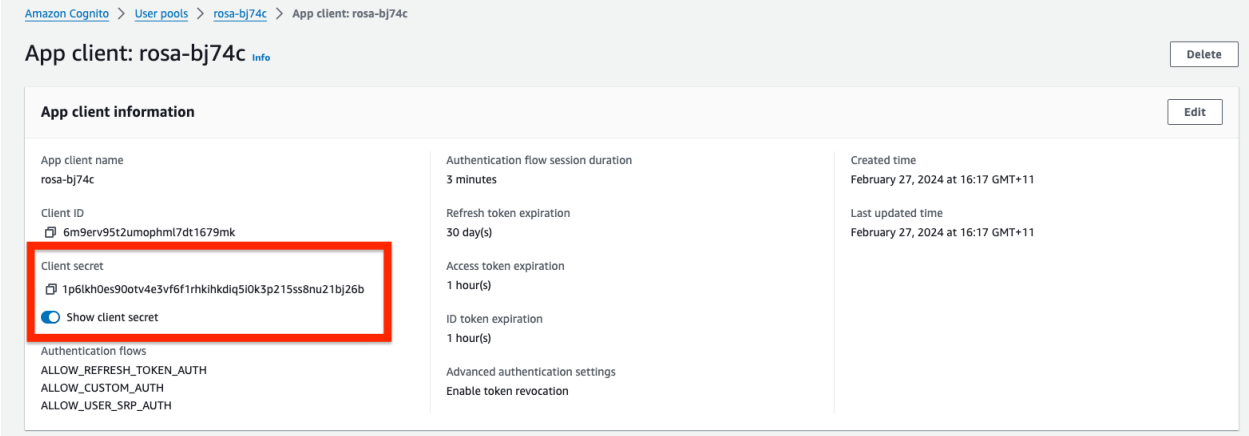
a. Set the ***Name*** field with a unique, identifiable value. Users see this name when they choose an available IDP.

b. Confirm that the ***OAuth callback URL*** field is correct. This field is prepopulated based on the cluster name, domain, and OpenID name.

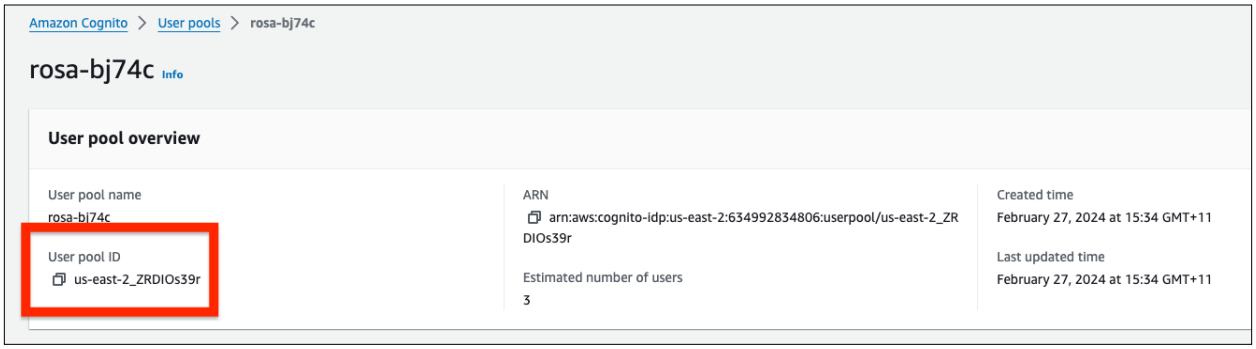
c. Set the ***Client ID*** field to the Amazon Cognito user pool client ID. Find this value in the ***App client information*** section of the Amazon Cognito console.



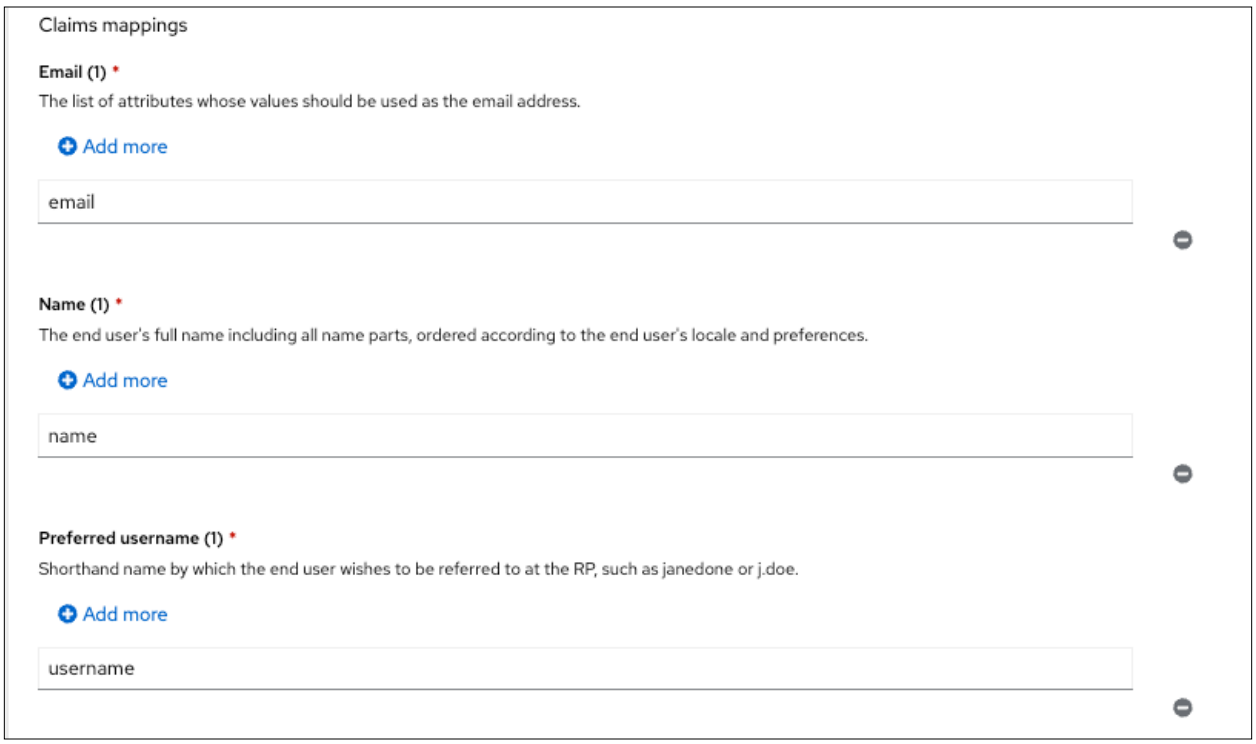
d. Set the ***Client secret*** to the Amazon Cognito user pool client secret. Find this value in the ***App client information*** section of the Amazon Cognito console.



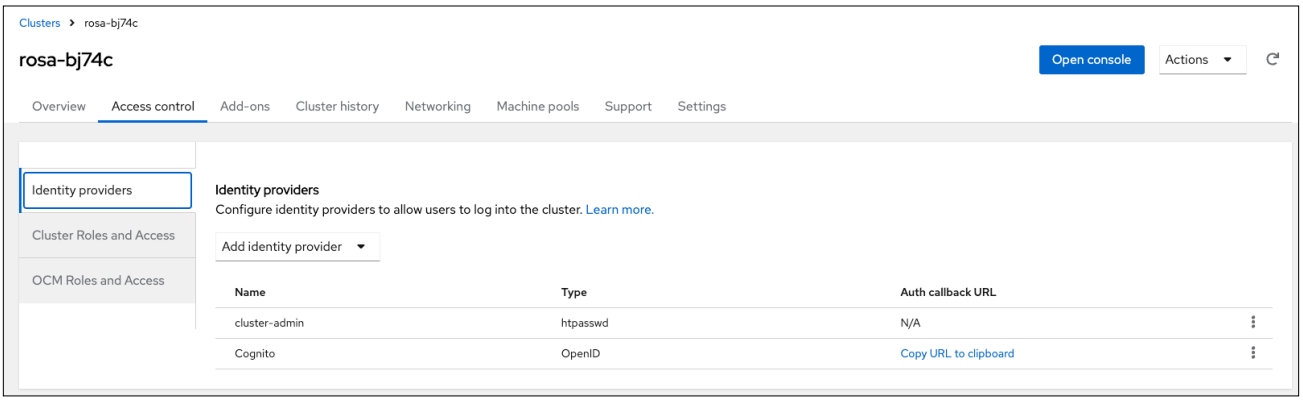
e. Set the ***Issuer URL*** field using the https scheme with no URL query parameters or fragments. The URL begins with ***cognito-idp***, followed by your AWS region, and then the user pool ID. For this example, the Issuer URL is ***https://cognito-idp.us-east-2.amazonaws.com/us-east-2_ZRDIOs39r***. You can find the user pool ID in the ***User pool overview*** section of the Amazon Cognito console.



f. Set the optional ***Claims mappings*** fields with email and username mappings.



6. View the status of IDPs in the ***Access control*** tab of Red Hat Hybrid Cloud Console.



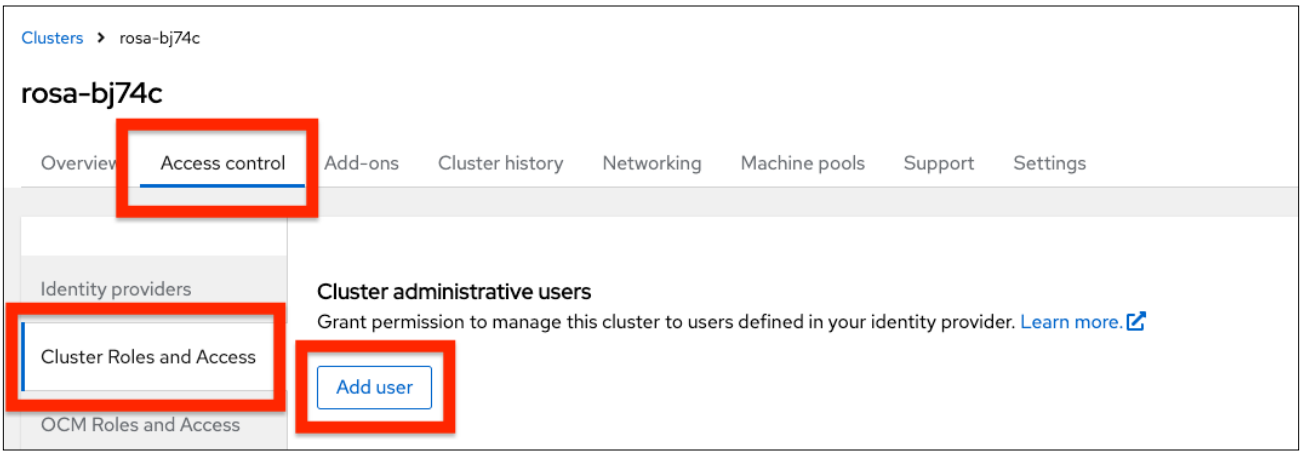
Set up OpenShift Service on AWS permissions on Amazon Cognito users

OpenShift Service on AWS uses RBAC to assign permissions to user objects. These objects determine the actions a user may perform. Administrators bind collections of rules—called roles—to user objects. Consult the Red Hat OpenShift RBAC documentation for more details.

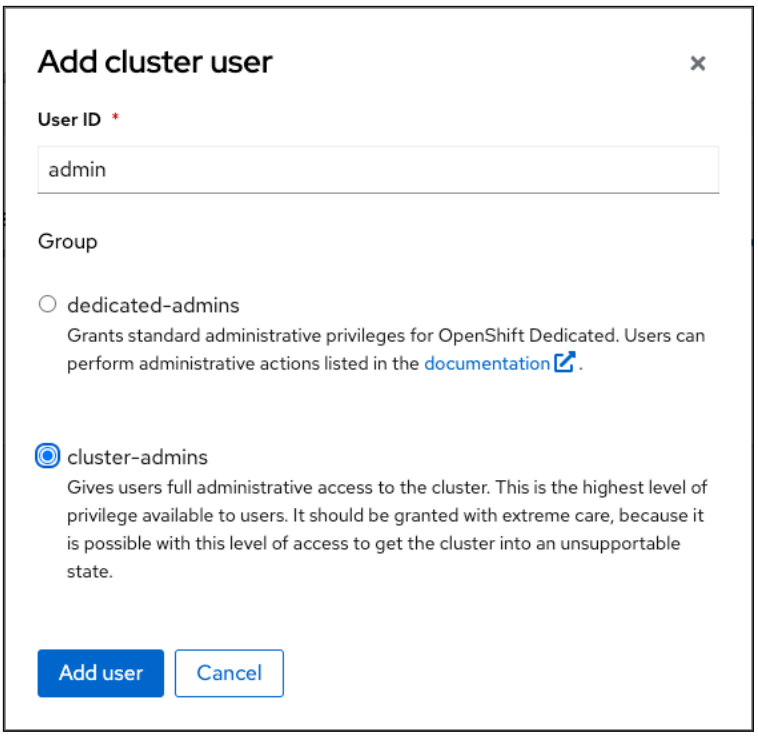
Follow these steps to add permissions for cluster administration to the Amazon Cognito admin user.

1. Navigate to ***Access control > Cluster Roles and Access*** in the appropriate cluster in **Red Hat Hybrid Cloud Console**.

2. Click ***Add user***.



3. Set the ***User ID*** field to ***admin*** and choose the ***cluster-admin*** group.

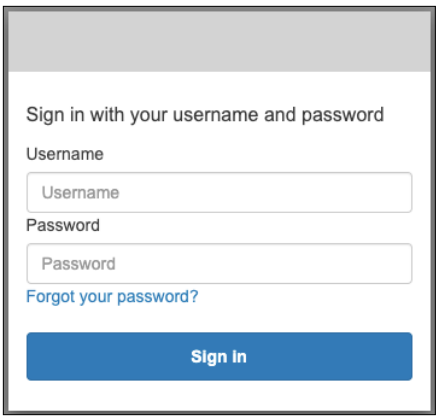


Log in to OpenShift Service on AWS using the Amazon Cognito IDP

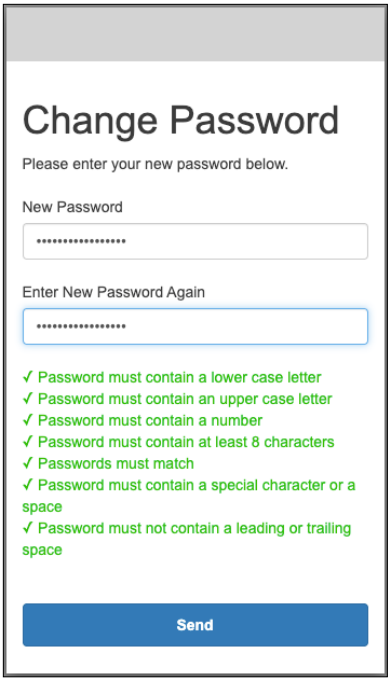
Follow these steps to log in and change your password.

1. Click ***Cognito*** on the OpenShift Service on AWS log-in screen to select the Amazon Cognito IDP.

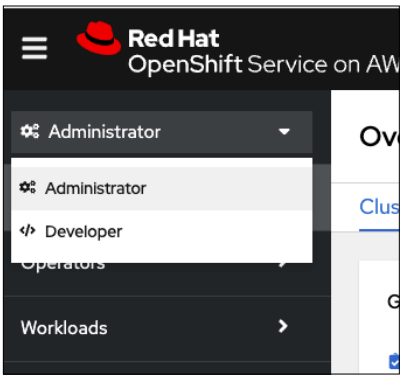
2. Provide your username and temporary password to authenticate with Amazon Cognito.



3. Change your password. Because this example selects ***Temporary password > Set a password*** when creating user accounts, Amazon Cognito requires all users to create a new password during their first login.



4. Confirm that the Amazon Cognito ***admin*** user can administer the cluster.



Chapter 6

Simplify log management and analysis with Amazon CloudWatch

By default, Red Hat OpenShift Service on AWS (ROSA) stores log data inside clusters, and understanding these metrics and logs is critical for successfully running your cluster. Included with OpenShift Service on AWS, the **Red Hat OpenShift logging operator** simplifies log management and analysis with centralized log collection, powerful search capabilities, visualization tools, and integration with **Amazon CloudWatch**—a monitoring and observability service from AWS. With this integration, you can collect, store, analyze, and visualize OpenShift Service on AWS infrastructure and audit and application logs directly in Amazon CloudWatch.

This chapter provides integration steps for both the CLI and the web console UI:

- Jump to **CLI instructions.**
- Jump to **Web console UI instructions**.

Integrate OpenShift Service on AWS and Amazon CloudWatch using the CLI

This section shows you how to use the CLI to integrate OpenShift Service on AWS and Amazon CloudWatch.

Gather information

Follow these steps to collect key information about your OpenShift Service on AWS and AWS environments.

1. Retrieve your AWS account ID and export it as an environment variable.

```
export AWS_ACCOUNT_ID=$(aws sts get-caller-identity \
  --query Account --output text)
```

2. Retrieve your cluster's OIDC URL by using the **oc** command to interrogate Red Hat OpenShift's authentication API and find the ***serviceAccountIssuer*** value. This property is the identifier of the bound service account token issuer. The OIDC URL is created when building your cluster using the OpenShift Service on AWS CLI. Consult the **documentation** for more details about OIDC verification options.

```
export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io \
  cluster -o json | jq -r .spec.serviceAccountIssuer | \
  sed  's|^https://||')
```

Prepare Amazon CloudWatch

Follow these steps to create the IAM policies and roles that allow Red Hat OpenShift service accounts to access Amazon CloudWatch.

1. Save the following policy to a file. This policy lets the service account create, view, and push to log groups and streams in Amazon CloudWatch.

```
cat << EOF > ${HOME}/policy.json
{
  “Version”: “2012-10-17”,
  “Statement”: [
     {
       “Effect”: “Allow”,
       “Action”: [
          “logs:CreateLogGroup”,
          “logs:CreateLogStream”,
          “logs:DescribeLogGroups”,
          “logs:DescribeLogStreams”,
          “logs:PutLogEvents”,
          “logs:PutRetentionPolicy”
         ],
       “Resource”: “arn:aws:logs:*:*:*”
     }
  ]
}
EOF
```

2. Create the policy. Filter and save the resulting Amazon Resource Name (ARN) in a variable.

```
POLICY_ARN=$(aws iam create-policy --policy-name “RosaCloudWatch” \
  --policy-document file:///${HOME}/policy.json --query Policy.Arn \
  --output text)
```

3. Ensure the variable is set correctly.

```
echo $POLICY_ARN
```

Here is a sample policy ARN. *123456789123* is the AWS account ID.

```
arn:aws:iam::123456789123:policy/RosaCloudWatch
```

4. Insert the AWS account ID and OIDC endpoint into the following trust policy and save it to a file. This policy specifies the trusted account members allowed to assume the CloudWatch role.

```
cat <<EOF > ${HOME}/cloudwatch-trust-policy.json
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
    "Federated":
"arn:aws:iam::${AWS_ACCOUNT_ID}:oidc-provider/${OIDC_ENDPOINT}"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "${OIDC_ENDPOINT}:sub":
"system:serviceaccount:openshift-logging:logcollector"
      }
    }
  }]
}
EOF
```

Here is a sample file that includes the AWS account ID and OCID endpoint. *1234567890123* is the AWS account ID, and *oidc.op1.openshiftapps.com/29abcdefghijklms46g* is the OIDC endpoint.

```
cat <<EOF > ${HOME}/cloudwatch-trust-policy.json
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
    "Federated":
"arn:aws:iam::1234567890123:oidc-provider/oidc.op1.openshiftapps.
com/29abcdefghijklms46g"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "oidc.op1.openshiftapps.com/29abcdefghijklms46g:sub":
"system:serviceaccount:openshift-logging:logcollector"
      }
    }
  }]
}
EOF
```

5. Create the role using the trust policy and replacing *<CLUSTERNAME>* with the name of your cluster. Filter the output and save the role ARN in a variable.

```
ROLE_ARN=$(aws iam create-role --role-name "RosaCloudWatch-<CLUSTERNAME>" \
  --assume-role-policy-document file://${HOME}/cloudwatch-trust-policy.json \
  --tags "Key=rosa-workshop,Value=true" \
  --query Role.Arn --output text)
```

6. Ensure the variable is set correctly.

```
echo $ROLE_ARN
```

Here is a sample role ARN.

```
arn:aws:iam::1234567890123:role/RosaCloudWatch-<CLUSTERNAME>
```

7. Attach the policy and role to use them with the service account.

```
aws iam attach-role-policy \
  --role-name <RosaCloudWatch-RoleName> \
  --policy-arn "${POLICY_ARN}"
```

Install OpenShift Service on AWS components

Follow these steps to install the Red Hat OpenShift logging operator.

1. Create an ***OperatorGroup*** for the logging system. An ***OperatorGroup*** provides multitenant configuration by selecting target namespaces in which to generate required access for member operators. This example creates an ***OperatorGroup*** called ***openshift-logging*** targeting the ***openshift-logging*** namespace.

```
cat << EOF | oc apply -f -
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-logging
  namespace: openshift-logging
spec:
  targetNamespaces:
  - openshift-logging
EOF
```

2. Create a *Subscription* for the operator. This contains operator installation, management, and lifecycle information, including the update channel, catalog sources, and installation namespace.

```
cat << EOF | oc apply -f -
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  labels:
   operators.coreos.com/cluster-logging.openshift-logging: “”
  name: cluster-logging
  namespace: openshift-logging
spec:
  channel: stable
  installPlanApproval: Automatic
  name: cluster-logging
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

3. Verify that installation is complete. Wait for this command to respond with ***deployment “cluster-logging-operator” successfully rolled out***.

```
oc -n openshift-logging rollout status deployment cluster-logging-operator
```

Grant access to Amazon CloudWatch

Secrets store sensitive information—API keys, passwords, and certificates, for example—that should not be exposed in the source code. By storing data in a secret, developers can keep critical information separate from the code, reducing the risk of accidental exposure. Secrets are typically referenced within applications using methods that ensure that sensitive data is only accessible to authorized components.

Create a secret that lets the OpenShift Service on AWS service account access Amazon CloudWatch. This example creates a secret called ***cloudwatch-credentials*** using the role ARN stored in a variable.

```
cat << EOF | oc apply -f -
---
apiVersion: v1
kind: Secret
metadata:
  name: cloudwatch-credentials
  namespace: openshift-logging
stringData:
  role_arn: ${ROLE_ARN}
EOF
```

Forward OpenShift Service on AWS logs to Amazon CloudWatch

ClusterLogForwarder custom resources let you send logs to third-party systems. With this resource, you can define both outputs and pipelines. An output is a destination for log data, while a pipeline is the routing from a log to an output. In this example, Amazon CloudWatch is an output.

ClusterLogging custom resources let you configure log collectors, storage, and visualization for the cluster. For this example, **fluentd** is the log collector, however other settings like **vector** are possible based on your logging substack.

Follow these steps to send logs from OpenShift Service on AWS to Amazon CloudWatch.

1. Create a ***ClusterLogForwarder*** custom resource. Observe that this custom resource uses many components created earlier, including the ***openshift-logging*** namespace and OpenShift Service on AWS secret.

```
cat << EOF | oc apply -f -
---
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
  - name: cw
    type: cloudwatch
    cloudwatch:
      groupBy: namespaceName
      groupPrefix: <CLUSTERNAME>
      region: $(aws configure get region)
```

```
    secret:
      name: cloudwatch-credentials
  pipelines:
  - name: to-cloudwatch
    inputRefs:
    - infrastructure
    - audit
    - application
    outputRefs:
    - cw
EOF
```

2. Create a ***ClusterLogging*** custom resource. This command instructs the logging system to begin forwarding logs to Amazon CloudWatch.

```
cat << EOF | oc apply -f -
---
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  collection:
    logs:
      type: fluentd
  forwarder:
    fluentd: {}
  managementState: Managed
EOF
```

3. Wait for OpenShift Service on AWS logs to arrive in Amazon CloudWatch.

```
watch aws logs describe-log-groups --log-group-name-prefix <CLUSTERNAME>
```

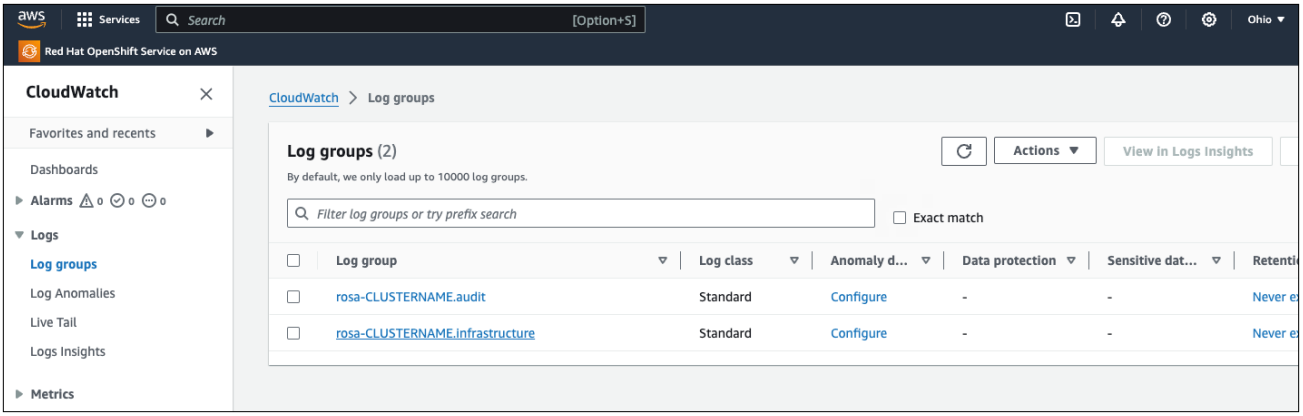
At first, you may see only empty log groups.

```
{
    “logGroups”: []
}
```

It may take up to 5 minutes for the log groups to begin filling.


```
{
  “logGroups”: [
    {
      “logGroupName”: “rosa-mkxrh.audit”,
      “creationTime”: 1710387473042,
      “metricFilterCount”: 0,
      “arn”:
“arn:aws:logs:us-east-2:1234567890123:log-group:rosa-CLUSTERNAME.audit:*”,
      “storedBytes”: 0,
      “logGroupClass”: “STANDARD”,
      “logGroupArn”:
“arn:aws:logs:us-east-2:1234567890123:log-group:rosa-CLUSTERNAME.audit”
    },
    {
      “logGroupName”: “rosa-mkxrh.infrastructure”,
      “creationTime”: 1710387437083,
      “metricFilterCount”: 0,
      “arn”:
“arn:aws:logs:us-east-2:1234567890123:log-group:rosa-CLUSTERNAME.
infrastructure:*”,
      “storedBytes”: 0,
      “logGroupClass”: “STANDARD”,
      “logGroupArn”:
“arn:aws:logs:us-east-2:1234567890123:log-group:rosa-CLUSTERNAME.infrastructure”
    }
  ]
}
```


4. You can also view the logs in the Amazon CloudWatch console.



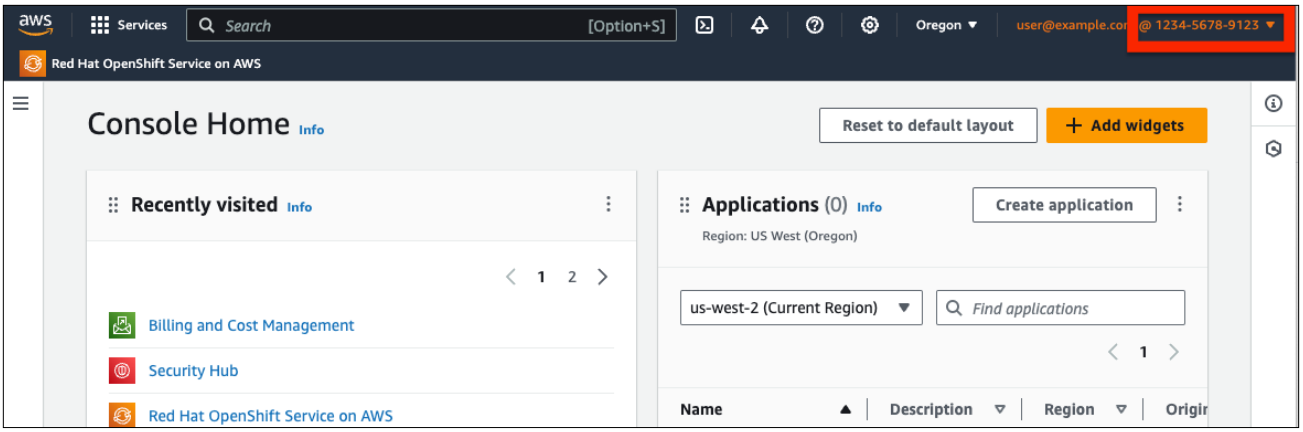
Integrate OpenShift Service on AWS and Amazon CloudWatch using the web console UI

This section shows you how to use the web console UI to integrate OpenShift Service on AWS and Amazon CloudWatch.

Gather information

Follow these steps to collect key information about your OpenShift Service on AWS and AWS environments.

1. Obtain your AWS account ID from the AWS Console.



2. Find your OIDC details using the CLI.

   Retrieve your cluster's OIDC URL by using the **oc** command to interrogate Red Hat OpenShift's authentication API and find the ***serviceAccountIssuer*** value. This property is the identifier of the bound service account token issuer. The OIDC URL is created when building your cluster using the OpenShift Service on AWS CLI. Consult the documentation for more details about OIDC verification options.

```
oc get authentication.config.openshift.io cluster -o json | \
  jq -r .spec.serviceAccountIssuer
```

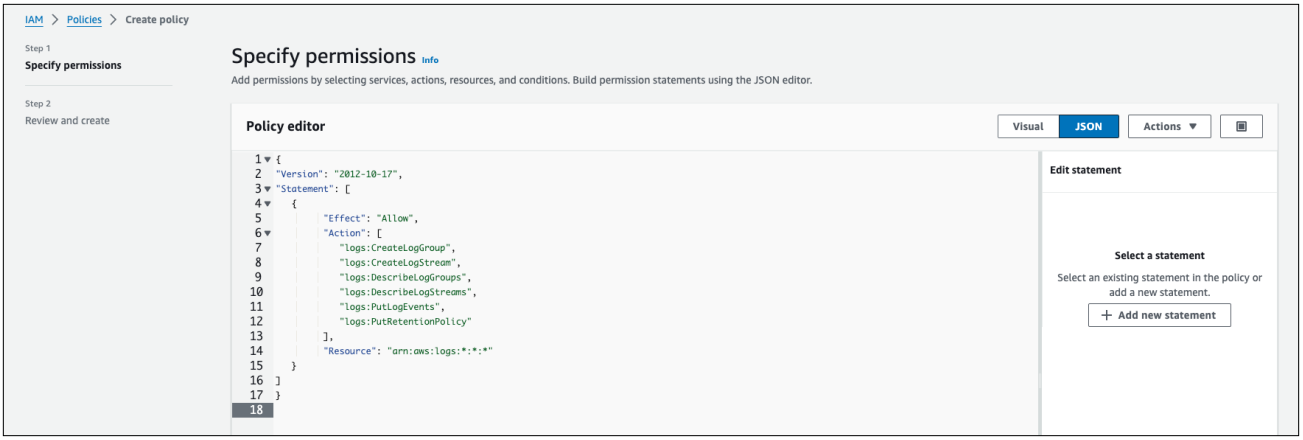
Prepare Amazon CloudWatch

Follow these steps to create the IAM policies and roles that allow Red Hat OpenShift service accounts to access Amazon CloudWatch.

1. Navigate to ***IAM > Policies*** in your AWS console and click ***Create Policy***.

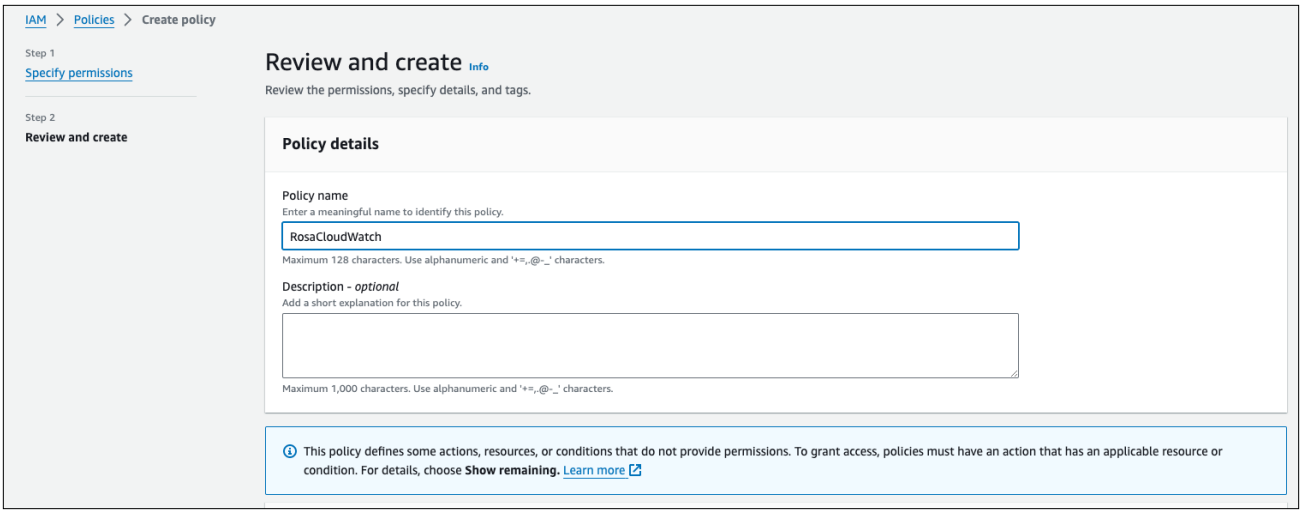
2. Select ***JSON*** and paste the following policy in the ***Policy editor*** window. This policy lets the service account create, view, and push to log groups and streams in Amazon CloudWatch.

```
{
  “Version”: “2012-10-17”,
  “Statement”: [
     {
       “Effect”: “Allow”,
       “Action”: [
          “logs:CreateLogGroup”,
          “logs:CreateLogStream”,
          “logs:DescribeLogGroups”,
          “logs:DescribeLogStreams”,
          “logs:PutLogEvents”,
          “logs:PutRetentionPolicy”
         ],
       “Resource”: “arn:aws:logs:*:*:*”
     }
  ]
}
```



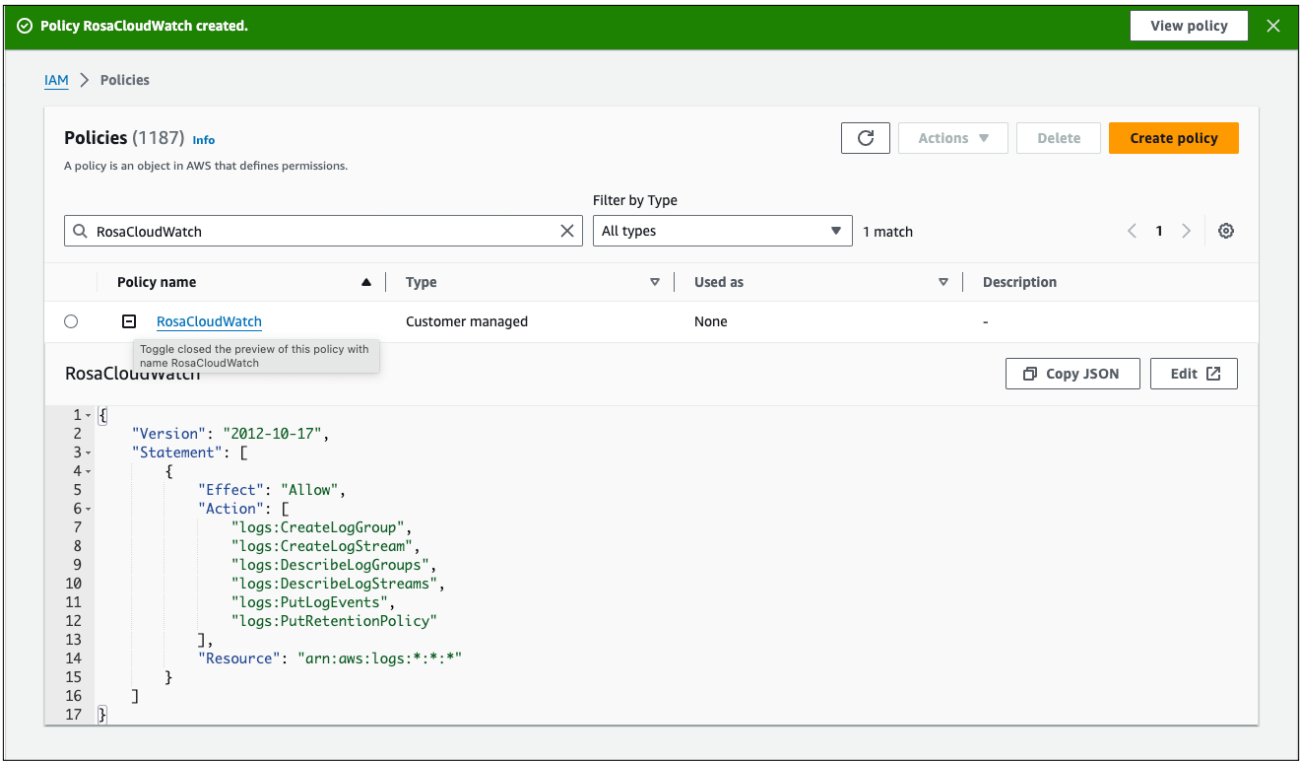
3. Click ***Next***.

4. In the ***Review and create*** form, set the ***Policy name*** field to ***RosaCloudWatch***.

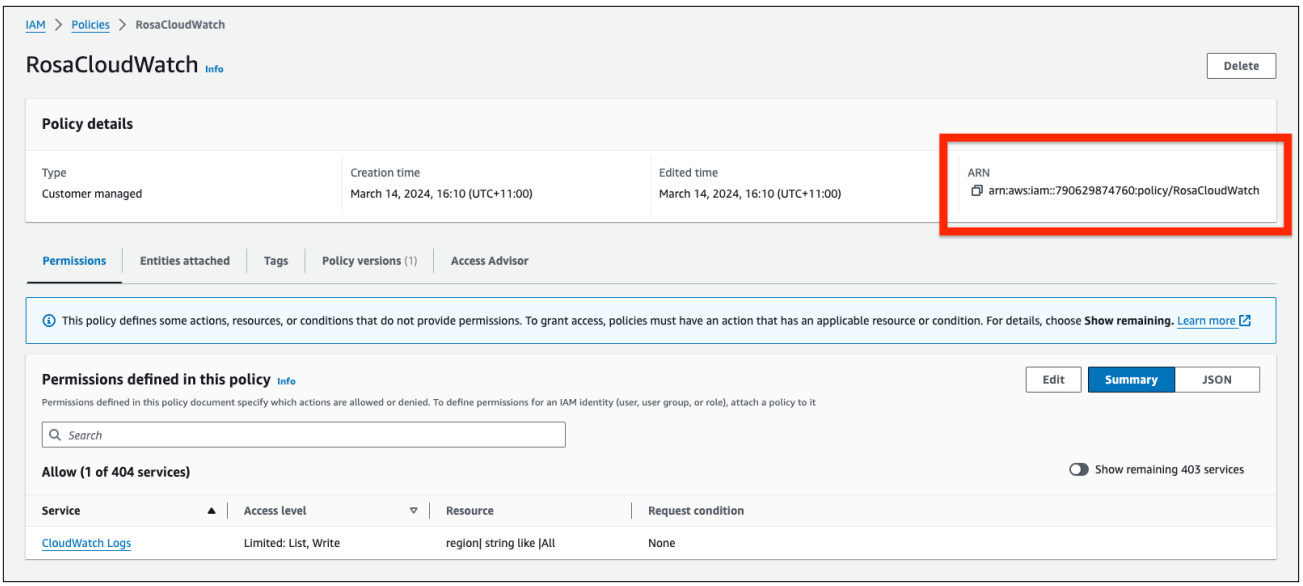


5. Click ***Create Policy***.

6. Search for the new ***RosaCloudWatch*** policy.

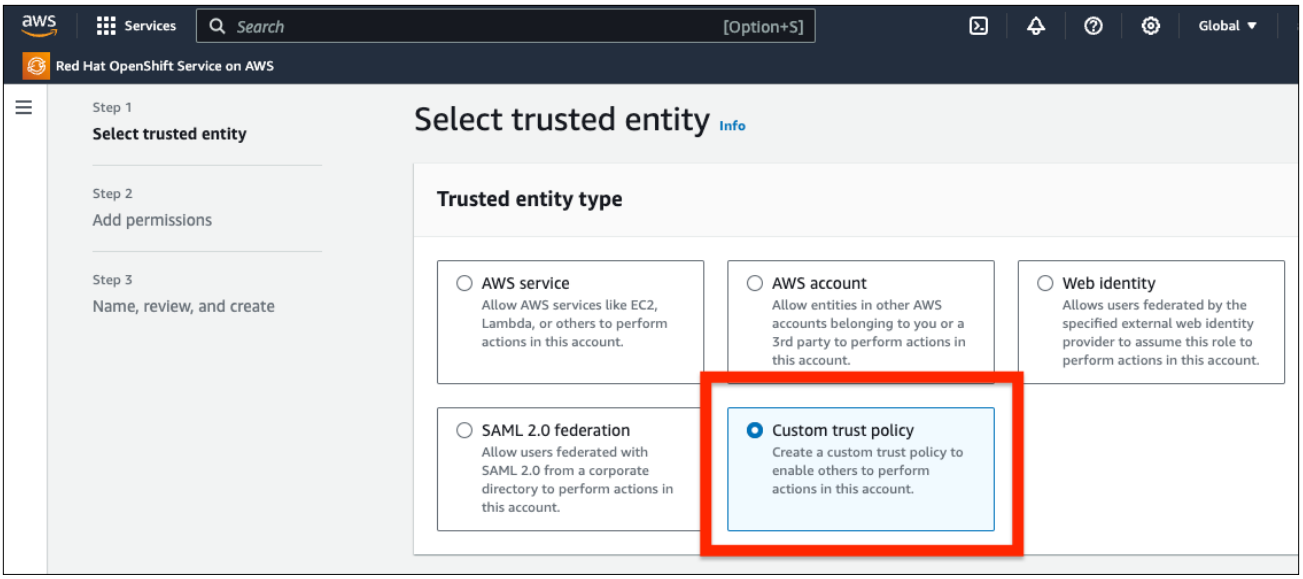


7. Click ***RosaCloudWatch*** policy. Find and record the ARN.



8. Navigate to ***IAM > Roles*** in your AWS console and click ***Create role***.

9. Select ***Custom trust policy***. This policy specifies the trusted account members allowed to assume the CloudWatch role.

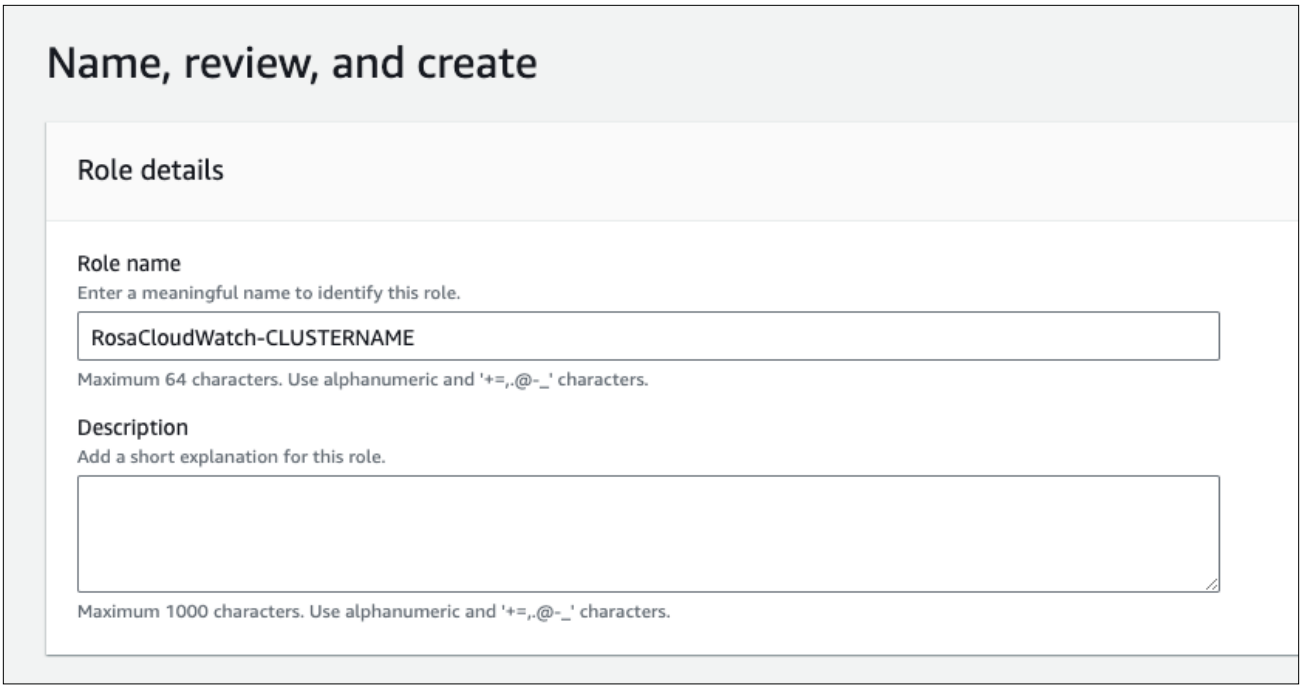


10. Insert the AWS account ID and OIDC endpoint into the following trust policy and paste it in the editing window.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
    "Federated": "arn:aws:iam::<AWS_ACCOUNT>:oidc-provider/<OIDC_ENDPOINT>"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "<OIDC_ENDPOINT>:sub":
"system:serviceaccount:openshift-logging:logcollector"
      }
    }
  }]
}
```

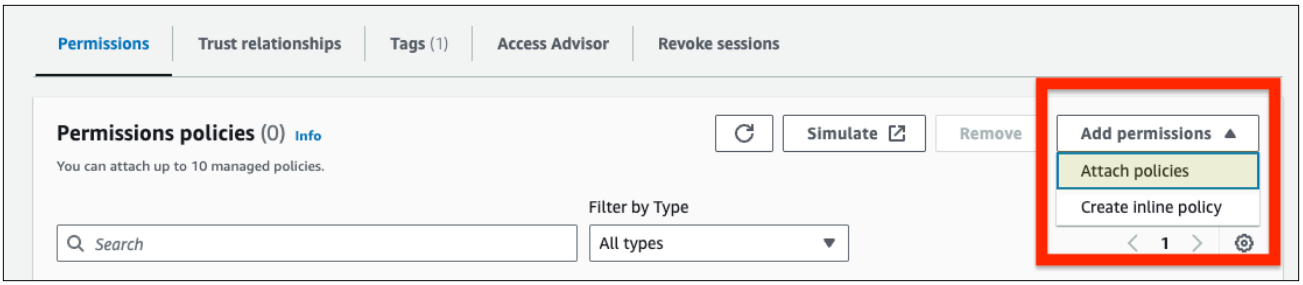
11. Click ***Next***.

12. Set the ***Role name*** field to ***RosaCloudWatch-<CLUSTERNAME>***, replacing ***<CLUSTERNAME>*** with the name of your cluster.

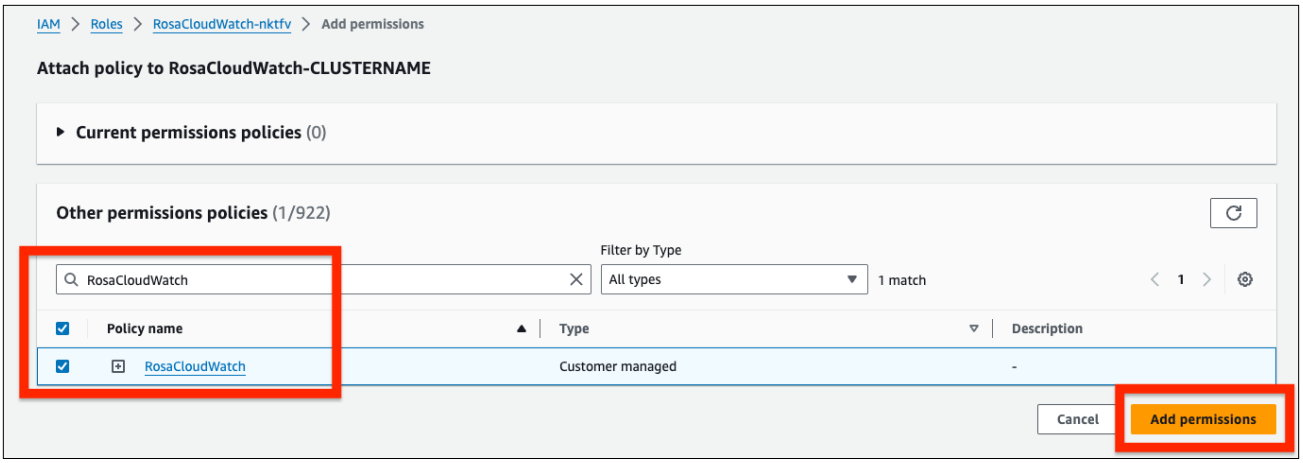


13. Navigate to the ***Permission policies*** section in the ***Permissions*** tab of the ***RosaCloudWatch-<CLUSTERNAME>*** role.

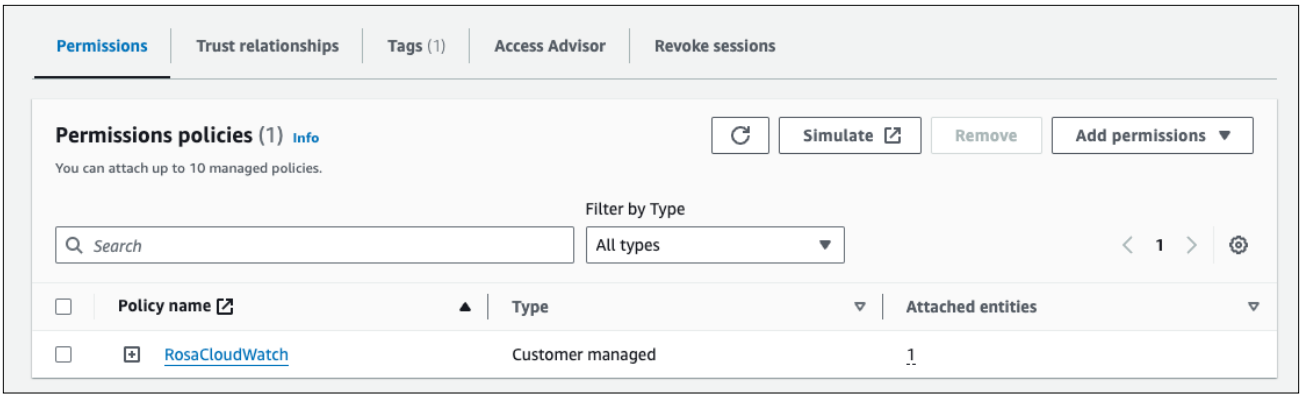
14. Choose ***Attach policies*** in the ***Add permissions*** menu to attach the ***RosaCloudWatch*** policy to the role.



15. Search for the ***RosaCloudWatch*** policy. Select it and click ***Add permissions***.



16. Confirm that the policy is now attached to the role.

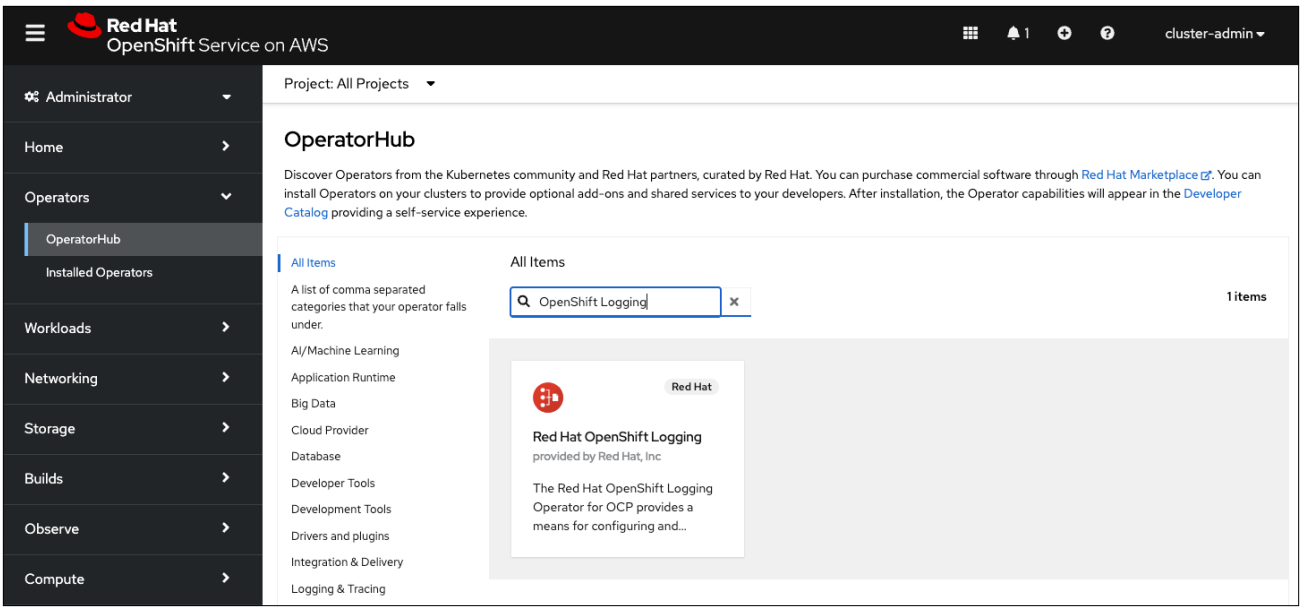


Install OpenShift Service on AWS components

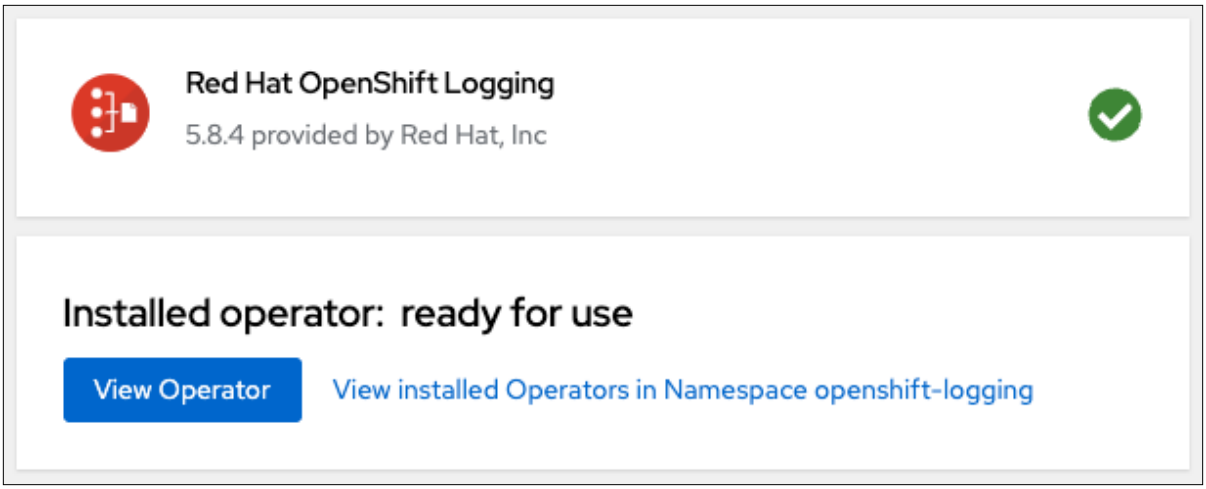
Follow these steps to install the Red Hat OpenShift logging operator.

1. Navigate to ***Operators > OperatorHub*** in the OpenShift Service on AWS console.

2. Search for ***OpenShift Logging***. Then click ***Red Hat OpenShift Logging***.



3. Accept the defaults and install the Red Hat OpenShift logging operator. Wait for the operator to report ***ready for use***.



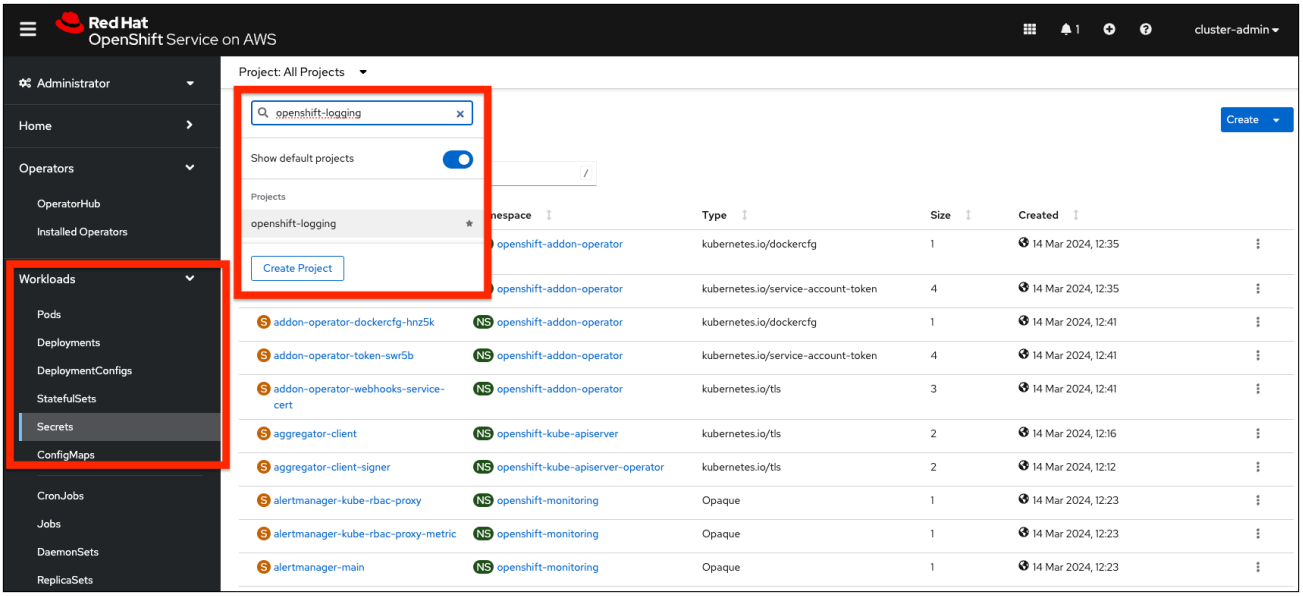
Grant access to Amazon CloudWatch

Secrets store sensitive information—API keys, passwords, and certificates, for example—that should not be exposed in the source code. By storing data in a secret, developers can keep critical information separate from the code, reducing the risk of accidental exposure. Secrets are typically referenced within applications using methods that ensure that sensitive data is only accessible to authorized components.

Follow these steps to allow the OpenShift Service on AWS service account to access Amazon CloudWatch.

1. Navigate to ***Workloads > Secrets*** in the OpenShift Service on AWS console.

2. Switch to the ***openshift-logging*** projects. You may need to toggle ***Show default projects***.



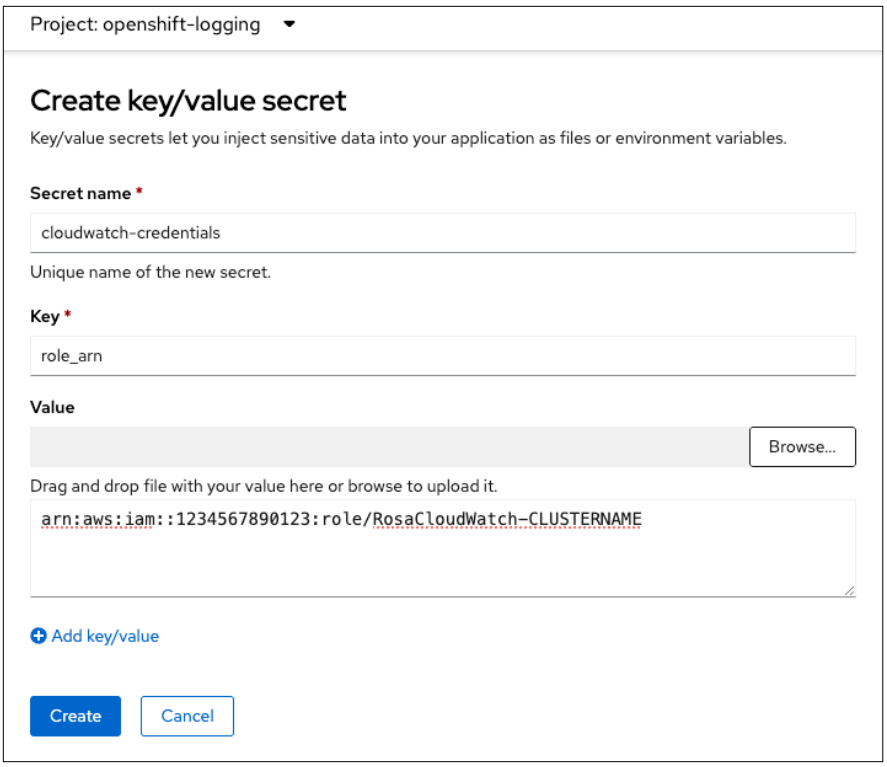
3. Choose ***Key/value secret*** in the ***Create*** menu.

4. Set the ***Secret name*** field to ***cloudwatch-credentials***.

5. Set the ***Key*** field to ***role_arn***.

6. Set the ***Value*** field to the ARN of the ***RosaCloudWatch-CLUSTERNAME*** role. For this example, the ARN is ***arn:aws:iam::1234567890123:role/RosaCloudWatch-CLUSTERNAME***.

7. Click ***Create***.



Forward OpenShift Service on AWS logs to Amazon CloudWatch

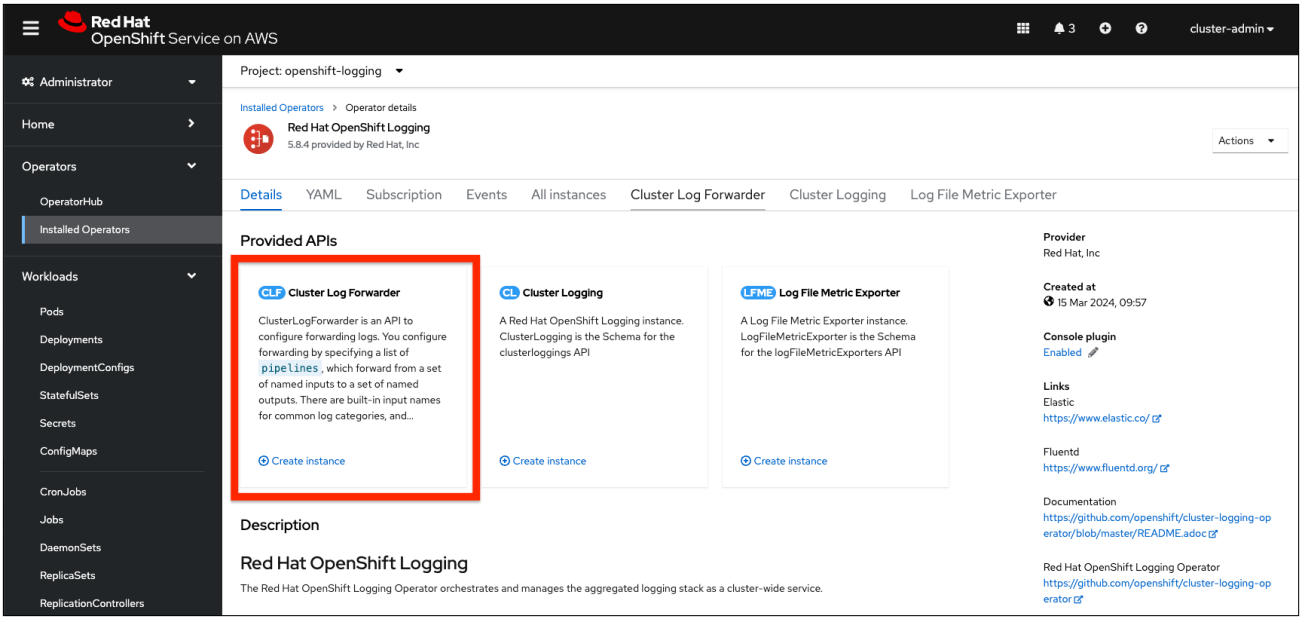
ClusterLogForwarder custom resources let you send logs to third-party systems. With this resource, you can define both outputs and pipelines. An output is a destination for log data, while a pipeline is the routing from a log to an output. In this example, Amazon CloudWatch is an output.

ClusterLogging custom resources let you configure log collectors, storage, and visualization for the cluster. For this example, **fluentd** is the log collector, however other settings like **vector** are possible based on your logging substack.

Follow these steps to send logs from OpenShift Service on AWS to Amazon CloudWatch.

1. Navigate to ***Operators > Installed Operators*** in the OpenShift Service on AWS console and select the ***Red Hat OpenShift logging operator***.

2. Navigate to the ***Cluster Log Forwarder*** tab and click ***Create instance*** under the ***Cluster Log Forwarder*** provided API.



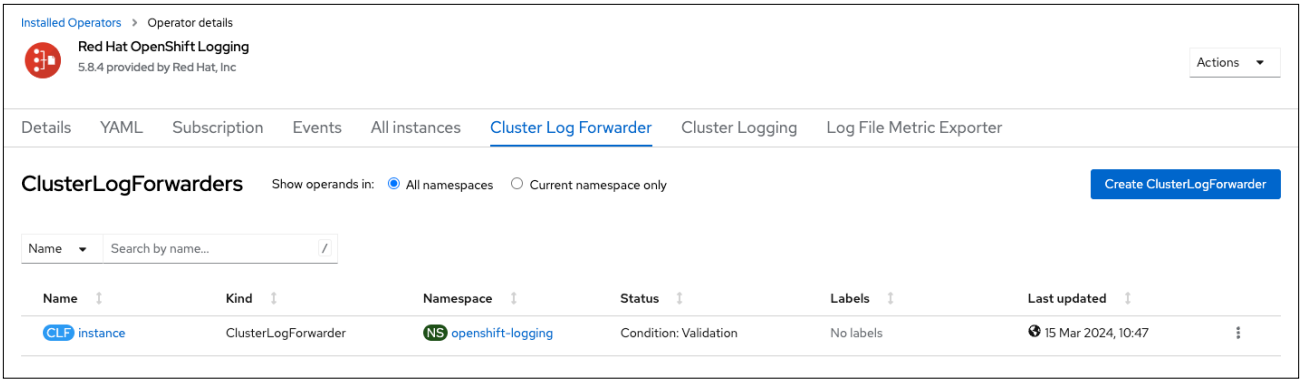
3. Select ***YAML view***. The ***Create ClusterLogForwarder*** UI lets you manually set all fields, however, it is easier to create this custom resource using YAML.

4. Update the following custom resource definition with your AWS region and OpenShift Service on AWS cluster name. Then paste it into the editing window.

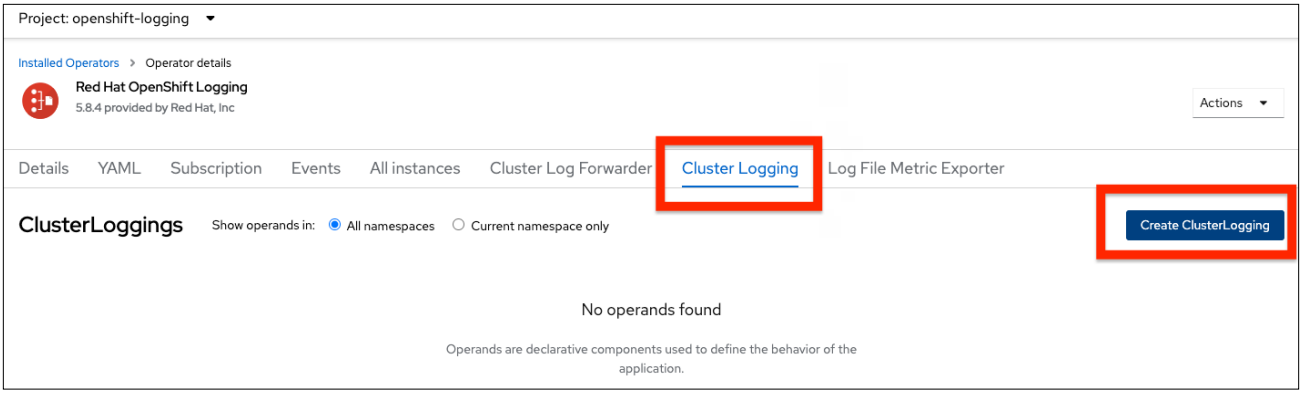
```
---
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
  - name: cw
    type: cloudwatch
```

```
    cloudwatch:
      groupBy: namespaceName
      groupPrefix: <CLUSTERNAME>
      region: <AWS_REGION>
    secret:
      name: cloudwatch-credentials
  pipelines:
  - name: to-cloudwatch
    inputRefs:
    - infrastructure
    - audit
    - application
    outputRefs:
    - cw
```

5. Click ***Create ClusterLogForwarder***. You can view the new resource in the console.



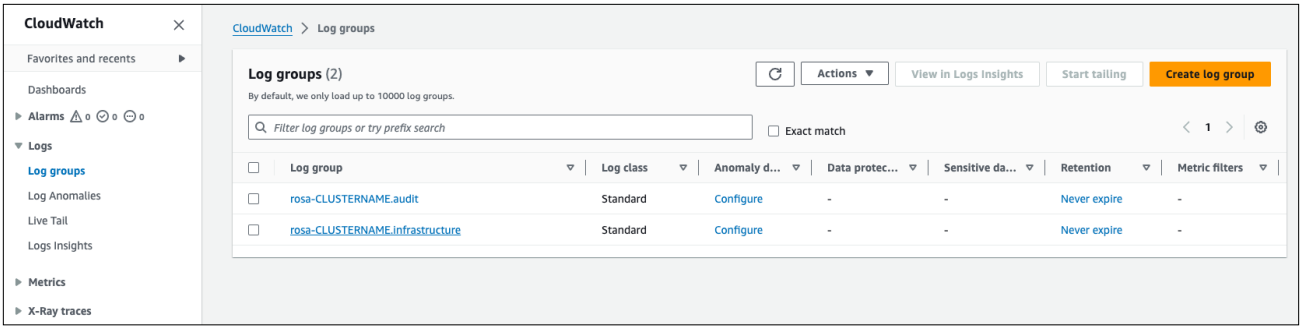
6. Navigate to the ***Cluster Logging*** tab and click ***Create ClusterLogging***.



7. Select ***YAML view*** and paste the following custom resource definition into the editing window.

```
---
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  collection:
    logs:
      type: fluentd
  forwarder:
    fluentd: {}
  managementState: Managed
```

8. Wait for the ***Status*** column to report ***Ready***.



9. You can now view your logs in the Amazon CloudWatch console.

Learn more

Red Hat OpenShift Service on AWS (ROSA) provides a fully managed application platform that lets you run Red Hat OpenShift clusters without worrying about the underlying infrastructure. As a first-party offering, OpenShift Service on AWS integrates efficiently with AWS services like Amazon Cognito and Amazon CloudWatch, allowing you to use your preferred tools and services. Read more about **OpenShift Service on AWS** and get started with a **free, 8-hour hands-on experience**.

For more information and support during your OpenShift Service on AWS evaluation, contact Red Hat and AWS.

0824_KVM